Department of Computer Science and Software Engineering
University of Canterbury

January 2008

Assessing the Impact of Positive Feedback in Constraint-Based Tutors

A thesis submitted in partial fulfilment of the requirements
for the Degree of Master of Science in Computer Science
in the University of Canterbury by

Devon K. Barrow

Supervisor: Professor Dr. Antonija Mitrović[1]
Associate supervisor: Dr Mick Grimley[2]
Examiner: Professor Dr. Benedict du Boulay[3]

[1]Department of Computer Science and Software Engineering, University of Canterbury

[2]Department of Educational Studies and Human Development, University of Canterbury

[3]Department of Informatics, School of Science and Technology, University of Sussex

Abstract

Across many domains, Intelligent Tutoring Systems (ITSs) are used to facilitate practice, providing a customized learning environment and personal tutoring experience for students to learn at their own pace through effective student modeling and feedback. Most current ITSs are built around cognitive learning theories including Ohlsson's theory on learning from performance errors and Anderson's ACT theories of skill acquisition which focus primarily on providing negative feedback or corrective feedback, facilitating learning by correcting errors. Research into the behavior and methods used by expert tutors suggest that experienced tutors use positive feedback quite extensively and successfully. This research investigates positive feedback; learning by capturing and responding to correct behavior, supported by cognitive learning theories. The research aim is to develop and implement a systematic approach to delivering positive feedback in Intelligent Tutoring Systems, in particular SQL-Tutor, a constraint-based tutor which instructs users in the design of Structured Query Language (SQL) database queries. An evaluation study was conducted at the University of Canterbury involving a control group of students who used the original version of SQL-Tutor giving only negative feedback and an experimental group using the modified version of SQL-Tutor where both negative and positive feedback were given. Results of the study show that students learn quite similarly from one system to another, however those in the experimental group take significantly less time to solve the same number of problems, in fewer attempts compared to those in the control group. Students in the experimental group also learn approximately the same number of concepts as students in the control but in much less time. This indicates that positive feedback results in increased amount of learning over a shorter period of time and improves the effectiveness of learning in ITSs.

Contents

List of Figures

List of Tables

Acknowledgments

The author would like to extend sincere thanks to his supervisor, Dr. Antonija Mitrović for her dedication, support and encouragement throughout the course of this project without which it would not be possible. He also wishes to thank his co-supervisor Dr. Michael Grimley whose valued input resulted in the increased quality and relevance of the research work performed. Thanks to Stellan Ohlson for his valued contributions to key areas of this research project; these have impacted the course of this work significantly. To all members of the ICTG group, for your support, sound advice and countless assistance, thank you! Finally, to my entire family, Joan and Dennis Armstrong my parents, Amobi my brother and Debbie my sister, who have supported me throughout my academic life, and to God, the rock on which I stand.

CHAPTER 1

Introduction

1.1 Intelligent Tutoring Systems Facilitating Learning

Intelligent Tutoring Systems (ITSs) continue to grow in popularity and application. These systems provide one on one tutoring at relatively low cost and have the added flexibility with regard to timing, location, and amount of the tutoring experience. ITSs offer customized feedback to each learner allowing them to learn at their own pace. It has been shown time and time again, that computerized tutors with a problem-solving environment are ideal tools for enhancing student learning and offer an effective supplement for classroom teaching. Evaluation of a cognitive tutor for writing programs in the LISP computer language showed that students in the experimental group completed problems in one third (1/3) the time of those under control conditions with improvement in learning of 1 standard deviation [Koedinger 1998]. Koedinger in [Koedinger 1998] reports on two different cognitive tutors for geometry proof design. Both were used in classroom studies and compared to control classes using a traditional geometry curriculum without the cognitive tutor. In both studies, students in the experimental classes scored 1 standard deviation (SD) better than students in control classes. Atlas, a tutoring system for teaching Physics, improves performance by 0.9 standard deviations

[Freedman 1999], while SQL-Tutor, an ITS for teaching the Structured Query Language (SQL), improves performance by 0.65 standard deviations in just two hours of interaction with the system [Mitrovic, Mayo, Suraweera and Martin 2001]. Evaluation studies of SHERLOCK [Lajoie and Lesgold 1989], a tutor for technical troubleshooting in avionics shows that between 20 and 25 hours of interaction with the systems is equivalent to four years on-the-job experience.

Improvements in the quality and effectiveness of tutoring provided by ITSs has resulted in increased use. The American Society for Training and Development (ASTD) has reported that in the largest companies, employee training time spent with "learning technologies" has been increasing by over 10% in 2001 [Ong and Ramachandran 2003]. According to the United States National Center for Education Statistics, the use of computers to support education and learning has more than doubled between 1984 (36.2%) and 1997 (84%) [Planty, Provasnik and Hussar 1999]. Intelligent tutoring systems and related research components are at the forefront of these technologies. There is however still considerable work to be done in making such systems more effective.

While a great deal of work has been done with understanding learning and modeling the student, the current direction towards improving ITSs and maximizing effective learning is developing a theory of tutoring which would link the various learning mechanisms to the types of information the learner needs and hence what the tutor (ITS) should provide. This approach is being merged with work being done in observing the typical behavior of expert tutors and effective tutoring, and looking at the types of information and strategies used in effective human tutoring. Current hypotheses propose that student learning is correlated with the tutor's use of knowledge-construction activities e.g. dynamic plan scaffolding, and refection/generalization techniques [Heffernan 2001]. There is a move to incorporate these techniques into ITSs through modeling of tutorial actions and strategies as observed with expert human tutoring [Heffernan and Koedinger 2002][Core, Moore, Zinn and Wiemer-Hastings 2000]. Positive feedback is one of the teaching strategies which continues to surface throughout tutoring protocols. Work being done with tutoring protocols at the Department of Psychology at the Uni-

versity of Illinois at Chicago [Ohlsson, Eugenio, Chow, Fossati, Lu and Kershaw 2007] shows extensive and effective use of positive feedback by experienced human tutors. A logical step for Intelligent Tutoring Systems (ITSs) therefore seems to be extending the proposed model of tutoring to incorporate positive feedback.

1.2 Feedback in Learning

At the very heart of the learning experience is the idea of feedback, information provided in response to a learner's actions. In [Ohlsson 2004] Ohlsson investigates how students learn and acquire knowledge both declarative and procedural and concludes that practice or repetition of a given task usually accompanied by improvement, indicates that instrumental knowledge (knowledge about how to attain goals, achieve effects or perform tasks) is being acquired. The acquisition of this instrumental knowledge is through practice and by means of feedback. Feedback helps learners in several ways including detection of errors, correction of faulty knowledge and misconceptions, creation of new knowledge directly or indirectly and producing flow [Kulhavy and Stock 1989]. Feedback has to be considered a very important and critical factor in the support of effective learning in Intelligent Tutoring Systems (ITSs) because such systems seek to replicate the benefits and effectiveness of human to human tutoring.

The history of feedback research is one that is long and well-documented (see [Kulhavy and Stock 1989] for an excellent review) both in psychology (e.g. [Thorndike 1913]) and computer science programmed instruction (e.g. [Skinner 1968]). Compared to recent research, most early research on feedback focused on positive feedback to strengthen correct responses and neglected errors which did not receive feedback and were weakened. This approach was driven primarily by behaviorist and associationistic views in which feedback was regarded as a contingent event serving to reinforce or weaken student responses. As a result, the focus of feedback was more formal and technical concentrating on the frequency and delay rather than the complexity of the feedback content. This mechanistic view of the way in which feedback works empha-

sized positive feedback, that is positive consequences for successful performance and helped move educators toward a more positive instructional stance [Mason and Bruning 2001].

It was quickly realized however that this approach was lacking a critical component of learning, error detection and correction, and the emergence of information-processing theory in the 1970s and 1980s sparked a new approach in cognitive science and psychology to address this deficiency. This new information-processing approach included the ACT-R theory [Anderson 1983] [Anderson, Corbett, Koedinger and Pelletier 1995] and recognized the hidden information in errors. It provided a systematic means for error correction, viewed as a mental process resulting in an improvement in the learners knowledge about the task. Today it represents the accepted theoretical basis for analyzing, developing and implementing feedback in computer-based instructional systems including ITSs (see reviews by [Mory 1995][Kulhavy and Stock 1989] for excellent discussions of information processing and earlier perspectives on feedback). The approach however appears to have neglected positive feedback altogether as most systems concentrate principally on learning by error detection and correction.

In this research we want to investigate the role positive feedback has in learning, thinking of positive feedback not simply as a form of exhortation but we hypothesize that positive feedback given under the right circumstances [when the student is guessing or exhibiting uncertainty] will improve learning.

1.3 Motivation for Research

As previously mentioned, ITSs have proved successful in a number of domains, including highly complex domains. Currently the best ITSs achieve 1SD, but this is lower than 2SD, the measure of improvement achieved by the average student obtaining human tutoring [Bloom 1984] and researchers continue the search for methods of group instruction which are as effective as one-to-one human tutoring. Continued research in Intelligent Tutoring Systems is towards improving the effectiveness of learning within

such systems. One way of achieving this is by making improvements in the existing architectures enhancing underlying modules. Another way is expanding on and integrating new ideas about learning and instruction. Because so much is still not known about the human brain and how exactly people learn and store knowledge, extensive research takes place utilizing the latter approach, that is, making and evaluating conjectures of learning and instruction.

While significant work has been done in the area of understanding, modeling and developing effective tutoring, including the development of a cognitive model of human tutors [Heffernan 2001] and comprehensive studies of expert versus non-expert tutoring [Heffernan 2001][Lu 2006], no work has been done with regards the systematic use of positive feedback, a tool quite often used by expert tutors. Instead, most studies have concentrated on tutor moves or techniques e.g. scaffolding, demonstrating, knowledge-construction and tutoring strategies such as those proposed in [Heffernan and Koedinger 2002] e.g. concrete articulation strategy and introduced variable strategy. ITSs including those built using constraint-based modeling and model tracing techniques, continue to be developed based on the principle of learning through error correction and while most are fully capable of doing so, provide only minimal feedback on correct responses.

This research proposes the development of a systematic approach to delivering positive feedback in Intelligent Tutoring Systems, in particular SQL-Tutor, a constraint-based tutor which provides instruction for users in the design of SQL database queries. It is hoped that this approach can be extended to other tutors to improve learning. This research seeks answers to the questions, how and why does positive feedback work to improve learning.

1.4 Overview of Research

Research into the behavior of, and methods used by expert tutors suggest that experienced tutors use positive feedback quite extensively and successfully. If current ITSs are to improve in performance and quality of learning experience delivered, they must

also include this widely used and successful form of feedback. In this research we investigate the reasons why positive feedback works.

It can be difficult and is quite often puzzling why positive feedback works. This type of feedback provided to the student on the basis of a correct response, sometimes contains no information or domain content e.g. “Great Job!” or “Well done, You should try the next question”. This information is often short, very simplistic and of no relevance to the material being delivered. Furthermore, such feedback by definition follows the correct actions of the student and as such would seem of nil effect, coming after the fact. Data however shows that experienced tutors do use it and use it quite effectively. We therefore ask ourselves why positive feedback works. What function does positive feedback play and where does it fall within learning theory?

Stellan Ohlsson proposes the following hypothesis[1] which we will investigate in this research project. The hypothesis suggests that most student steps are tentative, meaning that the student is guessing or is at least uncertain as to what to do. This is particularly so during the initial learning phase. Positive feedback therefore works by helping to remove or reduce the amount of uncertainty associated with student actions. Students are pressured to provide answers, to come up with solutions either constructively or to simply make it up. In either case, if such a move happens to be correct then providing assurance to the student of its accuracy helps to reduce uncertainty and apprehension [Ohlsson 2007]. With reduced uncertainty we expect to see greater strengthening of domain concepts and principles as compared to when only negative feedback is provided. It is the combination of error correction and strengthening which influences the rate of learning (producing the power law learning curve), not only the error correction mechanism.

This research answers several questions including when positive feedback should be given. We have identified a number of critical points within the tutoring experience where positive feedback can be given. These include:

[1] Private communication dated January 18, 2007

- When the student is expressing uncertainty but nevertheless does the right thing.
- When the student is too paralyzed to do anything at all.
- When the student has overcome aspects of the domain commonly agreed upon as being difficult and challenging.
- When the problem or task has been successfully completed and at major goals within the tutoring session.

Other issues investigated, include what content should be presented in the feedback message, how often feedback should be presented and how feedback should be presented (verbal feedback or text). These factors will help to define our approach to positive feedback in Intelligent Tutoring Systems (ITSs).

To evaluate our hypothesis, an evaluation study was conducted. Two versions of a Constraint-Based ITS (SQL-Tutor) were used in the study: a control version in which only negative feedback was given and an experimental version which gave negative feedback as well as positive feedback using the systematic approach to positive feedback developed through a combination of literature review and an analysis of tutoring protocols. Participants for the study where students from an undergraduate database course at the University of Canterbury, who used the system for the second term in 2007. Data such as pre and post-tests, student models, logs of student actions and instances of positive feedback were recorded and statistically analyzed. Learning curves [Mitrovic and Martin 2004] were plotted, results of data analysis interpreted and presented, and conclusions reached based on the initial hypothesis. A detailed account of the research is given in this research report.

1.5 Structure of Thesis

This thesis consists of seven chapters covering all aspects of the research topic. What follows next in chapter 2 is a summary of the background information and review of

literature related to this research including an overview of Intelligent Tutoring Systems and a review of the role feedback plays in learning and how this is facilitated by such systems. The chapter while not exhaustive, considers in some detail the factors influencing feedback in learning and focuses on the presentation of feedback, the content of feedback messages and the scheduling of feedback. In chapter 3 the goals and hypotheses of the research are outlined and some motivation for the project follows through a review of positive feedback in learning in section 3.3.

In chapter 4 the core work of the research is presented and the approach taken to positive feedback is considered in detail. Issues such as when and how often positive feedback should be given are addressed. How the content should be selected and what should be presented in the positive feedback message and the form and method of presentation are considered and together the answers and incite provide the guidelines for developing the approach to positive feedback. Chapter 5 contains the design and implementation of the research study, presenting in great detail implementation of the feedback approach in the chosen tutor, SQL-Tutor including generation of positive feedback content and the use of constraints to schedule delivery of positive feedback. A description of the evaluation including methods used and participants follows in chapter 6. The results of the evaluation is also presented from section 6.4 onwards, and detailed discussion and analysis of all findings given. Conclusions and findings are finally summarized in chapter 7 which also suggests improvements for the research and presents ideas for future work.

CHAPTER 2

Background

2.1 Intelligent Tutoring Systems

One-on-one human tutoring continues to be the ideal form of tutoring and has always been the choice for learning as psychological research has shown that it is more effective than traditional classroom instruction increasing students learning performance by two standard deviations [Bloom 1984]. A learning improvement of two standard deviations indicates that the student exposed to one-on-one tutoring, on average, performs as well as the top two percent (2%) of those obtaining classroom instruction. The goal of ITSs is to enable students to acquire deep, robust knowledge and obtain skills that can be used to solve different kinds of problems. In so doing, such systems hope to achieve results similar to those observed when being tutored individually by a human tutor.

Computers have been used for over 30 years in education. The first such systems to be deployed as an attempt to teaching using computers were Computer-based training (CBT) and computer aided instruction (CAI) [Beck, Stern and Haugsjaa 1996]. Instruction in these types of systems was not individualized and so did not cater to user's specific needs. Intelligent Tutoring Systems (ITSs) go beyond training simulations to answer specific user questions and to provide guidance on an individual basis. Unlike

other computer-based training technologies, ITSs assess each learner's actions within these interactive environments and develop a model of their knowledge, skills and expertise. Based on the learner/student model, the ITS tailors instructional strategies, in terms of both the content and style, and provide explanations, hints, examples, demonstrations and practice problems as needed.

A significant amount of current research is directed towards understanding and modeling expert human tutors. The pedagogical module is that part of the tutoring system which seeks to provide a model of this expert teaching process. Decisions about when to review, when to present a new topic and which topic to present are all made and affected by the pedagogical module. Studies such as [Heffernan and Koedinger 2002] include a model of tutorial reasoning incorporating the effective techniques that an experienced tutor uses.

In this research we will be using SQL-Tutor, an effective intelligent tutoring system that uses the constraint-based modeling approach to model student's knowledge.

2.1.1 Architecture of Intelligent Tutoring Systems

The first proposed ITS architectures consisted of four primary components: a curriculum module (the domain knowledge module), a student model, a tutor or pedagogical model, and an interface (communication module) between the student and the system [Nkambou and Kabanza 2001]. Since then, this basic architecture has been extended upon by many researchers, including [Nkambou and Gauthier 1996]; [Choquet, Danna, Tchounikine and Trichet 1998]; [Ritter, Brusilovsky and Medvedeva 1998]; [Frasson, Mengelle, Aimeur and Gouardéres 1996]; [Nkambou and Gauthier 1996] and [Nkambou and Kabanza 2001]. Figure 2.1 shows the basic architecture of an Intelligent Tutoring System (ITS).

Today's systems typically consist of a student modeller, domain module, pedagogical module and interface. Each of these modules provide information to other modules through a series of linkages. The student modeller stores information in the student

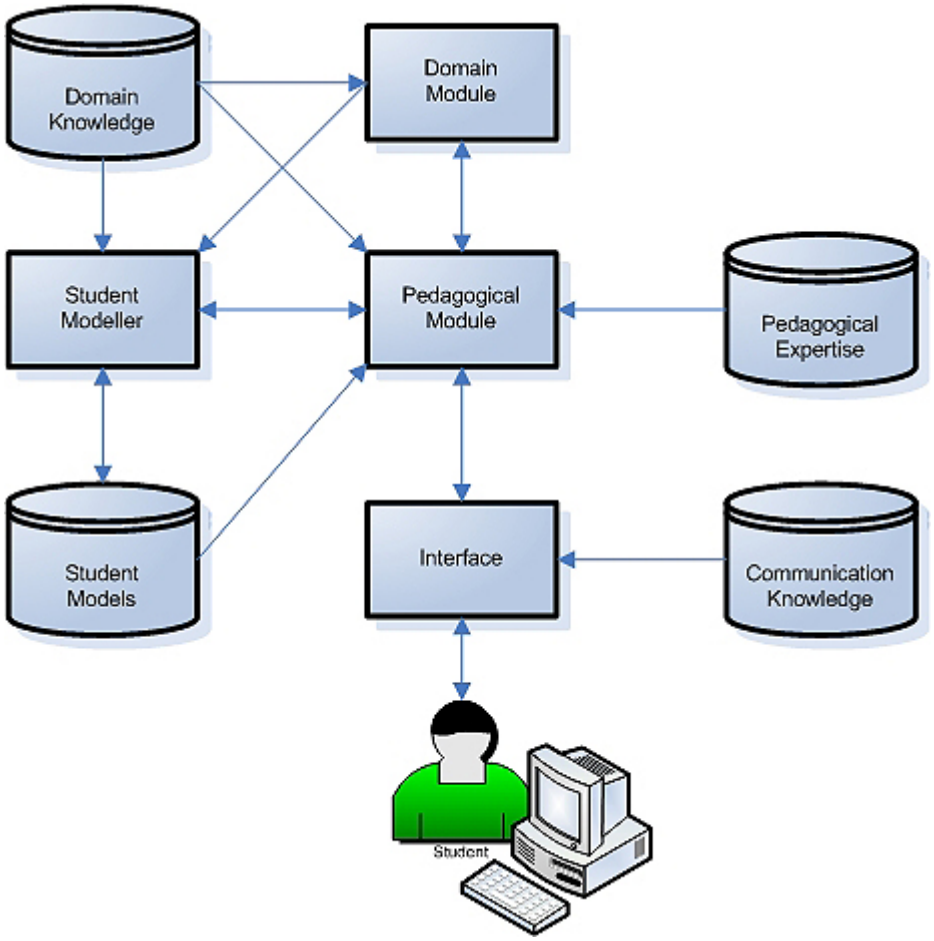

Figure 2.1: Architecture of Intelligent Tutoring Systems

model database that is specific to each individual learner. It obtains information from the domain module and processes this information together with information from the pedagogical module to provide user specific information back to the pedagogical module which in turn makes decisions about the tutoring process. The pedagogical module is at the center of the tutoring process and interacts with all components of the system feeding the results of teaching decisions to the interface which uses its communication knowledge to present information to the student including domain content and feedback messages. Below we present a more detailed discussion of each of these components.

The student model

The student model, as the model of a learner, represents the computer system's belief about the learner's knowledge. A student model can be defined as a representation of 'some characteristics and attitudes of the learner, which are useful for achieving the adequate and individualized interaction established between the computational environment' and the learner [Hartley, Paiva and Self 1995]. In order for an ITS to help a student learn a new concept, the ITS must maintain this detailed model of the individual student. A student model must contain information such as a description of the student's knowledge, learning abilities, strengths and weaknesses [Tsybenko 1995]. 'Building a student model involves defining who the leaner is including learner history and physical limitations; what is to be modeled e.g. the goals, plans, attitudes, capabilities, knowledge, and beliefs of the learner; how the model is to be acquired and maintained; and why information is being sought from/about the learner e.g. to give assistance to the learner, to provide feedback to the learner, or to interpret learner behavior' [Stauffer 1996]. The student model builds and maintains the understanding of the student. Student modeling is also viewed as a problem in its general form expressed as follows [Ohlsson 1993]:

> Given a behavioural record (of some sort), infer what the student knows and does not know about the relevant topic.

The student model must keep track of how well a student is performing on material being taught, the misconceptions made, the student's understanding of concepts and their acquisition and retention rate [Pillay 2003]. Many approaches both successful and unsuccessful have been taken to addressing this problem. The two most popular and widely used to date are the Constraint-based Modeling Approach (CBM) and the Model Tracing Technique (MT) [Ohlsson 1993]. Other approaches include machine learning and bug libraries. We describe both the CBM and MT approaches in later sections.

The pedagogical module

The pedagogical module decides on the teaching process based on the pedagogical knowledge it possesses and the student's state and history. Intelligent Tutoring Systems provide individualized instruction, one-on-one tutoring through the application of various teaching strategies and meta-strategies. Teaching strategies can be viewed as the individual methods and techniques employed to teach a particular concept whereas meta-strategies refer to the overall tutoring strategy utilized. Decisions made by the pedagogical module include but are not limited to which teaching strategy should be employed at any given point in the tutoring experience, the frequency with which feedback should be given and which topic should be presented next. Information from the student model is passed to the pedagogical module as input so that the decisions of the pedagogical module reflect the differing needs of each student effectively tailoring the tutoring experience to the particular student. The pedagogical module uses this information and based on the overall meta-strategy chooses the most appropriate teaching or instructional strategies and actions for the particular student and the topic at hand. Low-level issues the pedagogical module needs to consider as part of the meta-strategy include [Pillay 2003]:

- Which topic to present to the student?
- Which problems to present to the student?
- How frequently the student should be provided with feedback such as hints and error listings?

Because the tutor needs to know information about the student to be able to make these decisions, the pedagogical module relies heavily on the student modeling module.

The domain module

The domain module contains the knowledge and skills to be tutored. Generally, it requires significant knowledge engineering to represent a domain so that other parts of

the tutor can access it. One related research issue is how to represent knowledge so that it easily scales up to larger domains [Pillay 2003]. Approaches to the problem of knowledge representation include production rules which are a set of rules which match a student's steps in a traced performance of some task. These can be used to represent both correct and incorrect knowledge and provide increased modularity (single rules instead of entire procedures) and decreased grain size (single problem solving steps versus entire paths). ANDES, physics tutor is one example of such a system and contains approximately 600 rules [Schulze, Shelby, Treacy, Wintersgill, Vanlehn and Gertner 2000]. Constraint-based modelling based on Ohlsson's theory of learning from performance errors [Ohlsson 1996] uses the notion of state constraints to represent knowledge. Each constraint specifies a property of the domain that must be shared by all correct solutions and by so doing models correct knowledge explicitly and incorrect knowledge implicitly. A violated constraint signals an error caused by incomplete and incorrect knowledge. SQL-Tutor, an ITS for creation of SQL queries is an example of such a tutor and consists of 650 constraints dealing with syntactic and semantic errors [Mitrovic et al. 2001].

The communication module

The communication module contains both the interface and optionally, a stored representation of the communication knowledge. The general goal of the user interface is to use the available devices, usually keyboard, mouse, monitor, etc., to display to the student the necessary information, and use these devices to obtain student responses. Like the pedagogical module, the communication module deals with strategies, however these strategies are about the ways in which the system can communicate knowledge, primarily in the form of feedback to the student. It is charged with controlling interactions between the student and the system. Communication with the student must be engaged in a manner that reduces cognitive load while simultaneously providing the student with all information necessary for effective learning. The interface must clearly

indicate to the student where he or she is in a problem and errors made must be clearly visible to the student. Student history, learning style and other factors should influence the manner in which information is rendered for the student. The communication module is also responsible for informing the pedagogical module of the student's actions as he/she progresses through the course. Without this type of information it would be impossible to make decisions regarding the students interaction with the system and as such provide individualized tutoring.

Richard Mayer in his paper "Cognitive theory and the design of Multimedia Instruction" [Mayer 2002] notes that at the center of Cognitive theory lies three theory-based assumptions about how people learn from words and pictures. These are the

- Dual channel assumption
- Limited capacity assumption
- Active processing assumption

According to dual-channel theory, the mind processes visual and auditory content in separate memory systems. Text is initially processed visually, but it becomes an auditory element as the mind eventually hears the word [Mayer 2002]. The human cognitive system is seen as consisting of two distinct channels responsible for representing and manipulating knowledge: a visual-pictorial channel and an auditory-verbal channel. Pictures enter the cognitive system through the eyes and may be processed as pictorial representations in the visual-pictorial channel. Spoken words enter the cognitive system through the ears and may be processed as verbal representations in the auditoryverbal channel. When auditory and pictorial information are simultaneously presented, processing occurs independently and channels like highways, allow these two lanes of information into the brain. It is therefore better to present an explanation in words and pictures than solely in words [Mayer 2001]. Student with high spatial ability for instance have been shown to benefit from communication which includes pictorial representations. The second assumption is related to the limited capacity to remember.

Each channel in the human cognitive system has a limited capacity for holding and manipulating knowledge. When a lot of pictures (or other visual materials) are presented at one time, the visual-pictorial channel can become overloaded. When a lot of spoken words (and other sounds) are presented at one time, the auditory-verbal channel can become overloaded. The final assumption is that meaningful learning occurs when learners engage in active processing within the channels, including selecting relevant words and pictures, organizing them into coherent pictorial and verbal models, and integrating them with each other and appropriate prior knowledge. These active learning processes are more likely to occur when corresponding verbal and pictorial representations are in working memory at the same time.

The Cognitive Theory of Multimedia Learning is important to ITS development because most ITSs are deployed as multimedia applications using multiple forms of media for both input and output. The theory suggests several design principles to improve multimedia instructional design. These are as follow: Multimedia Principle, Contiguity Principle, Coherence Principle, Modality Principle, Redundancy Principle, Personalization Principle, Interactivity Principle and the Signaling Principle. These principles can be used in designing effective interfaces for ITSs, reducing cognitive load while allowing students to process as much information as possible. They will be further discussed in Section 4.3 and more information about these principles is available in [Mayer 2002] [Moreno and Mayer 2000].

2.1.2 Student Modelling

Intelligent Tutoring Systems (ITSs) are able to adapt to the different levels of user understanding and skill to produce learning effects. Student modelling forms a crucial part of this adaptation process and allows the system to gather information about learners knowledge level and problem solving skills including areas of weakness and strength, and also to track the learners progress and performance through the system including goal achievement and pattern of usage. Note that this is not an extensive listing of what

a student model possesses and as is usually the case, what an ITS models about a student will depend on the goal of the ITS, the domain and the pedagogical decisions that need making. Several techniques have been developed and used over time to generate such models. Some of these are complex and computationally expensive, for example Bayesian networks [Martin and VanLehn 1995] and the fuzzy logic approach [Kavcic 2002]. Bayesian networks use evidence of the student's interaction with the ITS to reason probabilistically about what a learner knows. The network consists of a series of nodes each providing a probabilistic measure of a student's knowledge of a given chunk of a domain. Fuzzy logic as described in [Kavcic 2002] uses an overlay approach where student knowledge is seen as a subset of domain knowledge. More specifically that approach uses an overlay over the domain model where the user model is a subgraph of the domain concept graph. Like the Bayesian approach, each node or knowledge chunk of this subgraph has a measure which explains the user's knowledge of that knowledge chunk. This measure is a triple of membership functions for three fuzzy sets of unknown, known and learned concepts. User knowledge can therefore be represented as a fuzzy graph with fuzzy relations and fuzzy nodes. Other techniques while computationally inexpensive take a considerable amount of time and expert knowledge to generate domain and student models and have limitations in what they can model. Below we will explore two of the more popular and widely used approaches to student modeling, model tracing and constraint-based modeling.

Model Tracing

Model Tracing is an approach to student modeling which involves trying to relate the behavioral manifestations of the student's solution on the computer to some sequence of production firings in the cognitive model [Anderson et al. 1995] [Anderson 1983]. This approach is used to model short-term knowledge while knowledge tracing (KT) is used to model long-term student knowledge [Anderson and Corbett 1994]. Model Tracing systems possess a computational model capable of solving the problems that

are given to students in the ways students are expected to solve the problems. In Model Tracing (MT) the student is monitored step by step for each problem and each step is modeled by identifying a rule capable of explaining the reason for the student's action. This rule is chosen from a library of all relevant correct and incorrect rules. Model Tracing was developed from the Advanced Computer Tutoring (ACT) theory initially the ACT* theory [Anderson 1983] and now the ACT-R theory [Anderson and Corbett 1994]. The theory explains a cognitive skill as consisting in large part of units of goal-related knowledge and the acquisition of such knowledge involves the formulation of thousands of rules relating task goals and task states to actions and consequences. This goal-oriented knowledge is represented by a production-rule formalism in which production rules take the following form:

> If the required goal is <goal> and the current situation is <situation>, then perform<action>

More specifically, consider the example of a geometry proof generation rule which might look as follows:

> IF the goal is to prove two triangles are congruent THEN set as a subgoal to prove that corresponding parts are congruent

or a LISP programming rule:

> IF the goal is to get the second element of the list and you have coded car THEN pass to car an argument that is the tail of the list

The theory makes a distinction between two kinds of knowledge; declarative knowledge e.g. knowing the fundamental theorem of arithmetic and procedural knowledge e.g. the ability to ride a bike. It proposes that the student primarily through instruction and observation receives and encodes goal-independent declarative knowledge (knowledge acquisition) which is then converted into production rules (a representation of a student's procedural knowledge). The student using various techniques e.g. comparison

and inference, generates problem solving behaviour by relating declarative knowledge to task goals, that is, practicing. These techniques and other interpretive problem solving procedures are converted into production rules using a learning process referred to as knowledge compilation. Finally strengthening is achieved through practice which produces smoother, more rapid, and less errorful execution [Anderson et al. 1995].

The computational complexity involved is equivalent to that of pattern matching and requires that an observed situation-action pair be matched to an existing rule that applies in that given situation. This simply means that the student's action taken in the current situation must match to some production rule whose situation is the same and whose specified action is equal to that taken by the student. The idea is that a production rule which matches the students relevant situation and results in the relevant action will be a possible explanation for what the student has decided to do in that situation. If a rule is missing then it is assumed that the students action was incorrect. Therefore a missing rule may result in incorrect student modeling and as a result incorrect pedagogical actions being taken. From this observation you should note that it is possible for a correct solution to be rejected. This is also the case with buggy rules, rules representing incorrect student actions. We mentioned above that rules are chosen from a library of both correct and incorrect rules (buggy rules). Buggy rules match student's incorrect actions and try to account for all the ways in which students can misunderstand or misconstrue a problem task. It is an extensive description of all false knowledge within a given domain. There is an obvious problem with this approach in that, that "the universe of mal-knowledge is incomprehensibly more vast than the universe of correct knowledge and the inference problem correspondingly more complicated" [Ohlsson 1994]. Building such rules, including rules representing correct knowledge, often requires extensive empirical research and is very time consuming and error prone.

Once the libraries have been built however, the process of matching can begin using some inference algorithm. The result of this will be a short term student model which will consist of a the set of rules which together explain or match the steps made in the traced performance including rules from possible different strategies.

Constraint-based Modeling

Constraint-Based Modeling (CBM) was first proposed as an approach to student modeling by Ohlsson [Ohlsson 1994], as a solution to overcoming the intractable nature of student modeling. The approach was successfully implemented and tested and since then has been used to develop a number of ITSs [Mitrovic and Ohlsson 1999]. In his theory of learning from errors [Ohlsson 1996] Ohlsson outlines a process of learning from errors, a process consisting of two phases: error detection and error correction. In order to first detect and recognize an error a student needs declarative knowledge. When declarative knowledge possessed by the learner is not encoded or correctly encoded to procedural knowledge, errors may occur. This supports the fact that even when someone has been taught the correct way to perform a task, mistakes are still made. The solution to this being practice works because it allows us to detect these mistakes ourselves or with assistance, and as a result of identifying these mistakes we encode correctly the appropriate rule(s) that had previously been violated. The foundation of CBM is the observation that the common property shared by all correct solutions to a problem is that they do not violate any of the basic principles of the domain. Unlike model tracing, CBM is concerned not with the exact sequence of states in the problem space the student has traversed, but places importance on the current problem state. The fundamental concept underlying constraints is that a correct solution cannot be derived by traversing a problem state which violates the fundamental principles of the domain. Errors are signified by violated constraints.

A state constraint is an ordered pair (C_r, C_s), where C_r is the relevance condition and Cs is the satisfaction condition. A solution is correct if it satisfies C_s for all constraints whose relevant conditions C_r are met. The members of this ordered pair can be thought to constitute a set of features or properties unique to a given problem state. The semantics of a constraint can therefore be viewed as [Mitrovic and Ohlsson 1999]:

> If $< relevance\ condition >$ is true, then $< satisfaction\ condition >$ had better also be true, otherwise something has gone wrong.

Let us consider the following example of a constraint in the addition of fractions domain:

> If $(x+y)/d$ is given as the answer to $x/d1+y/d2$, then it has to be the case that $d = d1 = d2$ (or else there is an error)

It does not matter what sequence of steps it was which brought the student to this state. Given that the student is in this state, that is, the student has written the solution to the fraction addition in common denominator form with no change to the value of the individual numerators x and y, then d must be equal to denominators d1 and d2. This means that it is incorrect to add two fractions by adding their numerators if they do not have the same denominator. In this example, the relevance condition C_r , is the complex predicate $(x+y)/d$ *is given as the answer to* $x/d1+y/d2$ and the predicate $d = d1 = d2$ is the condition that must be satisfied.

The process of testing whether a given problem state is consistent with a set of constraints becomes computationally equivalent to that of pattern matching and is can be summarized as follows: The student solution is matched to the relevance conditions of all constraints. The constraints whose relevance condition match the student solution are considered satisfied if and only if the student solution satisfies the satisfaction condition C_s. Violated constraints signify errors that violate fundamental concepts of the domain.

Constraint-based modeling significantly reduces the amount of processing required for the student modeling process. Unlike other student modeling techniques, Constraint-based modeling does not require a runnable expert module, which is difficult to build for many domains and compared to the model tracing approach, requires no extensive bug libraries to enumerate students' misconceptions about the domain [Mitrovic and Ohlsson 1999]. Moreover, since CBM evaluates the problem state rather than the path taken to arrive at the state, it stands robust in the face of creative solutions from students as well as inconsistent problem solving strategies.

2.2 Structured Query Language

The Structured Query Language (SQL) is the most popular computer language used to create, retrieve, update and delete data from relational database management systems. The language has evolved beyond its original purpose to support object-relational database management systems, and has been standardized by both the America National Standards Institute (ANSI) and the International Standards Organization (ISO) [Elmasri and Navathe 2007].

As a result of its popularity and its commercial relevance, SQL is widely taught at the tertiary level. Skill in writing SQL queries is a fundamental outcome expected by industry from any university course in Database Management Systems. SQL while a fourth generation programming language and designed to reduce programming effort, still poses quite a bit of difficulty for students. While the syntax has been standardized, students often have difficulty with fundamental SQL concepts such as multitable joins (join conditions), aggregation (difference between aggregate and scalar functions) and grouping. Errors also occur because of limited short term memory and the burden of having to memorize database schemas. Incorrect table names and attributes result in incorrect solutions and misconceptions in the student's understanding of the elements of SQL and the relational data model in general are what account for most SQL errors.

Some common student errors include missing the join conditions from a multi-table query or using non-aggregate, non-grouped attribute in a grouped query. For example many will express "find all books written by an Australian" by:

```
SELECT book.title
FROM book, author
WHERE author.nationality = 'Australian'
```

and

SELECT book.title

FROM book

GROUP BY book.authorname

Students face even greater difficulty when attempting to learn the language through proprietary RDBMS systems because feedback is limited to syntax only, see Figure 2.2 below and compare the feedback given by INGRES [Mitrovic and Ohlsson 1999] to that of SQL-Tutor. Semantic support is often provided in the form of online tutorials or books which do not provide support for practical trials by the student.

Example: For each director, list the director's number and the total number of awards won by comedies he or she directed if that number is greater than 1.

Correct solution:

```
SELECT DIRECTOR,SUM(AAWON)
FROM MOVIE
WHERE TYPE='comedy'
GROUP BY DIRECTOR
HAVING SUM(AAWON)>1
```

Student's solution:

```
SELECT DIRECTOR,SUM(AAWON)
FROM DIRECTOR JOIN MOVIE
ON DIRECTOR=DIRECTOR.NUMBER
WHERE TYPE='comedy'
```

INGRES: E_USOB63 line 1, The columns in the SELECT clause must be contained in the GROUP BY clause.

SQL-Tutor:

- You do not need all the tables you specified in FROM!
- You need to specify the GROUP BY clause! The problem requires summary information.
- Specify the HAVING clause as well! Not all groups produced by the GROUP BY clause are relevant in this problem.
- If there are aggregate functions in the SELECT clause, and the GROUP BY clause is empty, then SELECT must consists of aggregate functions only.

Figure 2.2: Inadequacy of Feedback given by Proprietary RDBMS [Mitrovic and Ohlsson 1999]

SQL-Tutor is aimed at bridging this gap providing an interface which allows feedback on a semantic level while providing a practical problem solving environment for the student.

2.3 SQL-Tutor

SQL-Tutor[1] is an intelligent tutoring system developed to assist university-level students acquire the knowledge and skills necessary to create SQL queries. It is designed as a practice environment with the prerequisite that students be previously exposed to the SQL concepts in lectures. Work on SQL-Tutor began in 1996, and now there are about several versions of the system available including the standalone version (see figure 2.3) and the web-enabled version, both developed in Allegro Common Lisp, Franz Inc. SQL-Tutor was first evaluated in 1998 (see [Mitrovic and Ohlsson 1999]) as an effort to design a constraint-based tutor for SQL, the database language. Since then the tutor has undergone a number of improvements and been evaluated through several studies from 1999 to the more recent 2006 study which investigated the effectiveness of problem templates on learning in intelligent tutoring systems [Mathews 2006].

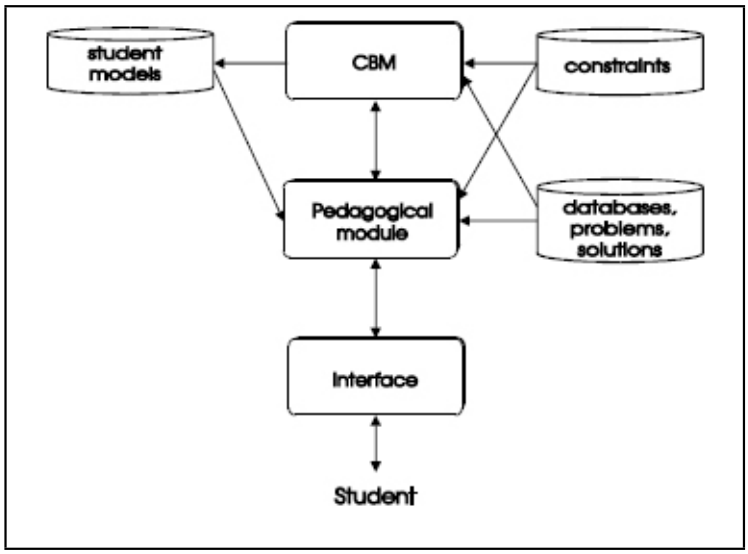

Figure 2.3: The Architecture of the Standalone Version of SQL-Tutor [Mitrovic and Ohlsson 1999]

The system currently covers only the SELECT statement of SQL, but the same approach could be used with other SQL statements. This focus on the SELECT statement

[1] SQL-Tutor available at http://ictg.cosc.canterbury.ac.nz:8000.

does not reduce the importance of the system, because most student misconceptions are caused by queries. Moreover, many of the concepts covered by SELECT are directly relevant to other SQL statements and other relational database languages in general [Mitrovic 1998]. The system, like most other intelligent tutoring systems, is not a substitute for the conventional style of education, but a complement to it.

This research will utilise the Web-enabled version of SQL-Tutor. In this version each student is assigned their own web session and the tutor selects a problem for the student to work on. Students submit solutions which are sent to the student modeller by the pedagogical module for analysis. The student modeller identifies any errors or mistakes and updates the student model accordingly to reflect student progress within the domain. Throughout our research we will take advantage of a number of features only available through the use of constraint-based tutors, SQL-Tutor being the first of such tutors. This includes the fact that constraint-based tutors are capable of keeping a history of all satisfied constraints. A detailed discussion of SQL-Tutor can be found in [Mitrovic 1998] [Mitrovic 2003].

2.3.1 Architecture

SQL-Tutor is implemented in Allegro Common Lisp (ACL), the first version being made available on SUN workstations and PC compatibles and subsequently on PC compatibles only. The tutor consists of an interface, a pedagogical module that determines timing and content of pedagogical actions and a constraint-based student modeller which analyzes and tracks how well students are performing. It contains definitions for several databases and a set of problems along with the ideal solutions for those problems. Students may work their way through a series of problems for each database, or ask the system to select a problem on the basis of his/her student model.

This study is based on the web-enabled version (Figure 2.4) which is based on HTTP/1.1 compliant Open Source web server AllegroServe, capable of hosting static and dynamic pages. AllegroServe is designed as a module that can be loaded into an

application to provide a web-based user interface to the application. It is therefore the optimal platform for a web-based ITS. The client-side of SQL-Tutor or its interface can be viewed in any common web browser as a set of dynamic HTML pages hosted by the web server. The main page of the interface contains a navigational frame, feedback frame and submission frame.

Communication between the interface and the server is built entirely on HTTP requests and responses exchanged by the web browser and the web server. Depending on the task the requests are initiated either by the applet or the HTML forms embedded in the pages. The interface makes requests using either GET and POST methods. As intended by the HTTP protocol, POST method is used for sending data to the server and GET method is used for retrieving information from the server. For example, when a user clicks on the Submit Answer button to submit a solution, a POST request delivers the users solution to the server and at the same time a GET request is initiated for updating the feedback frame.

To check the correctness of the student's solution, SQL-Tutor compares it to the correct solution, using domain knowledge. SQL-Tutor uses over 650 constraints to represent domain knowledge and compares submitted solutions to these constraints to determine its correctness. The student model passes information to the pedagogical module which then generates the appropriate feedback. In the case where the solution is correct and the problem has been solved or the student requires a new problem to work on, the pedagogical module uses the information from the student modeller to select an appropriate problem.

The core architecture of the system including the interface, remains primarily the same as the standalone version and this is what we will discuss below.

The Interface

The student is first greeted by a login page where they are required to supply a username and password (Figure 2.5). If it is a first time login, a new student model is created and

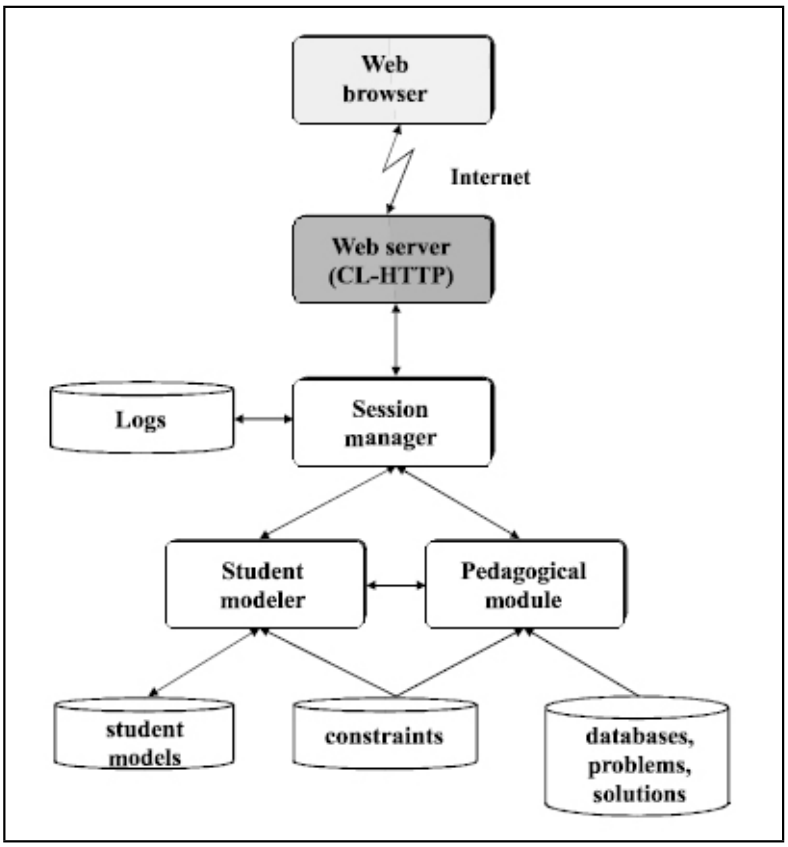

Figure 2.4: The Architecture of SQLT-Web, Web-enable Version of SQL-Tutor [Mitrovic 2003]

an initial screen gives information about how to use the system. Once the student successfully logs into the system, their student model is retrieved and the tutoring session effectively begins. The main window of the student interface is split into three distinct sections or frames.

The top part of the window consists of the main menu where the student has easy access to amongst other things help information, student model visualization and tutoring history. The upper part also displays the text of the current problem for easy reference. A listing of the clauses of the SELECT statement is displayed in the middle of the window outlining the structural elements of the SELECT statement. This reduces memory load because students need not remember the exact keywords nor the relative order of the clauses. The lowest part displays the schema of the currently cho-

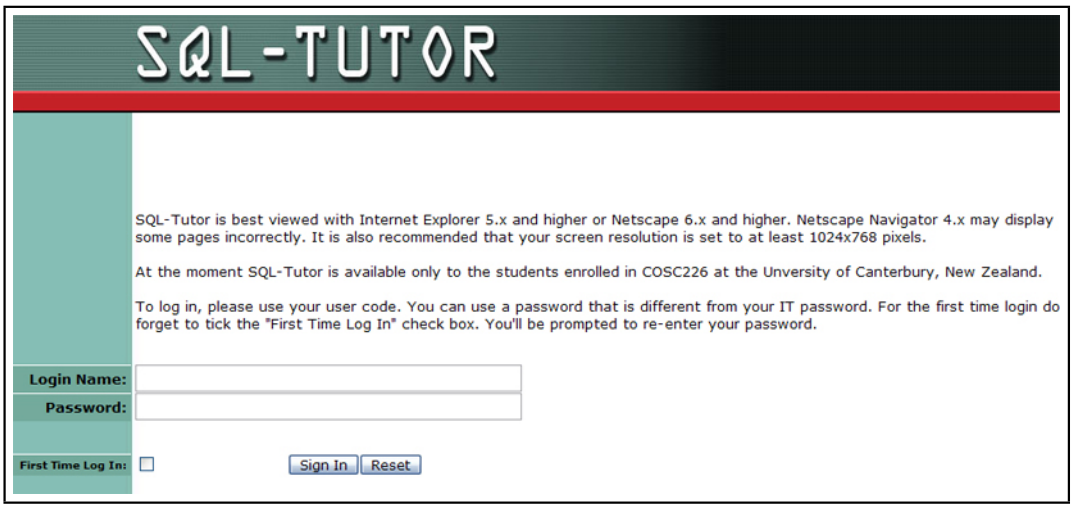

Figure 2.5: SQL-Tutor Login Screen

sen database. At the bottom is a display of the current database schema with helps to reduce cognitive load considerably. Displaying the database schema reduces the load required to constantly remember and reference table and attribute names and their semantics and checking low-level syntactic details. As a result the student is able to focus on the higher-level aspects of query definition.

The Student Modeller

The initialization of SQL-Tutor involves the creation of two structures from the compilation of semantic and syntactic constraints. These are called the relevance and satisfaction networks and they help to improve the excecution speed and efficiency of processing constraints. To check the correctness of the student's solution, SQL-Tutor compares it to the correct solution that is, the stored ideal solution and this results in a list of constraint violations. This is a two-step process.

The student's solution and the corresponding ideal solution are first propagated through the relevance network. This produces a list of relevant constraints meaning that their relevance conditions match the current solution state. This generated listing of constraints is then used as input in the second step. In the second step, the satisfaction components of the relevant constraints are compared to the student's current

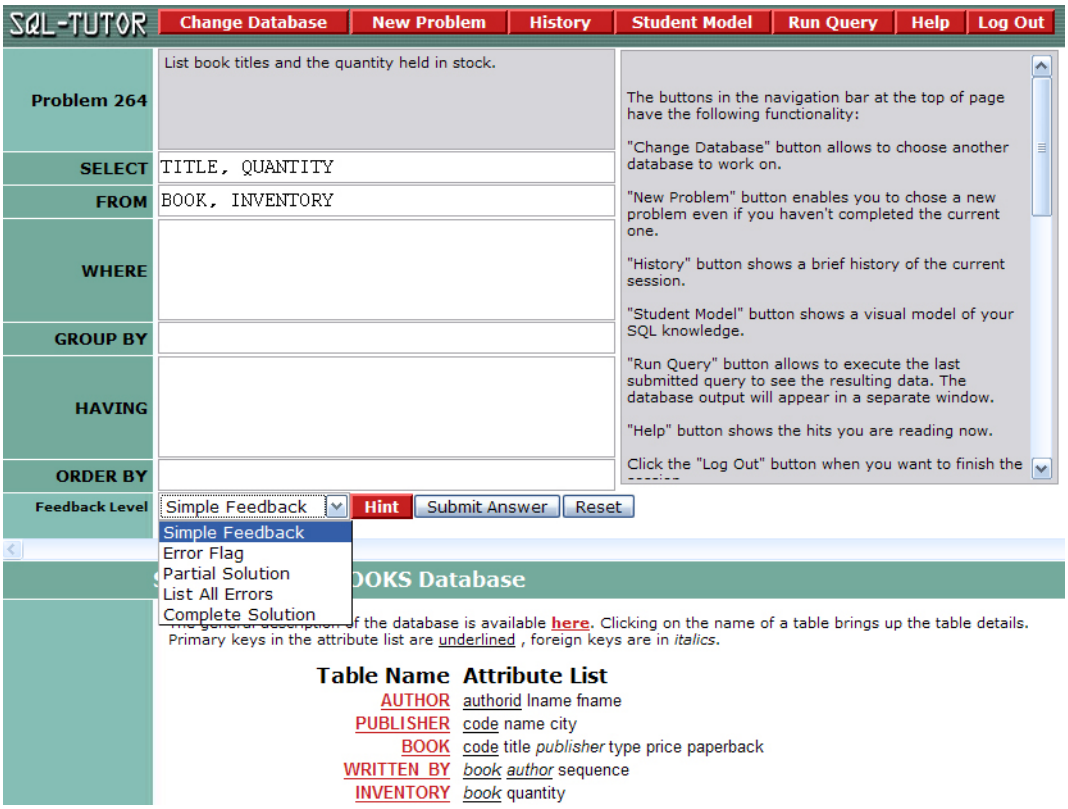

Figure 2.6: The Interface of SQLT-Web, Web-enabled Version of SQL-Tutor

solution state. This is accomplished by propagating both the students solution and the ideal solution through the satisfaction network. A constraint is said to be satisfied if the satisfaction condition matches either the student's solution or ideal solution or a combination of both. If the satisfaction condition fails to match then the constraint is said to be violated. The history of constraint usage is then stored in the student model.

The student model can be made to record extensive parameters however, it commonly records information about how often a constraint was relevant for both the ideal and student solution and how often it was satisfied or violated. This information is accumulated in three indicators, called relevant, used and correct. This record is used by the pedagogical module [Mitrovic and Ohlsson 1999].

The Pedagogical Module

In SQL-Tutor like in most other ITSs, the student model is used as input to the pedagogical module to produce individualized results. The pedagogical module uses this student model information to generate feedback messages and also to select SQL query problems that exercise particular learning components and as a result test the student's acquisition of a particular skill. This is done based on the student's level of understanding and domain knowledge.

Feedback SQL-Tutor tracks typical student behaviour and considers variations of correct and incorrect answers. It is able to identify errors in student conceptual thinking and points out the inferences that lead to the error occurring in the first place, that is, SQL-Tutor provides error specific feedback. It does not evaluate a students solution until it has been submitted and as such provides no feedback unless the student has explicitly submitted their problem solution. When the student submits a solution, it propagates through the relevance network as previously described. It is possible for a students solution to violate one ore more constraints. SQL-Tutor selects one constraint from a list of all violated constraints and provides feedback for this violated constraint. Selection of this constraint is based on the student model which contains a history of each constraint including the times when that constraint was relevant and satisfied or violated. Constraints are ordered and the constraint chosen is the first one to be violated. The feedback message is generated for only one constraint because it is assumed that it is easier for the student to work out one error at a time rather than deal with multiple errors and constraints thus producing multiple feedback messages which might be too overwhelming. If a student has violated a constraint several times then it shows a degree of faulty declarative knowledge, incorrect knowledge encoding or execution regarding that particular concept and so it would be appropriate to target that concept for remedial action.

This remedial action is taken in the form of feedback. SQL-Tutor has a default policy which starts with a minimal feedback level and goes upwards providing more information on each successive submission. The ITS is able to process each step and decide how much feedback to give based on this step-by-step analysis. SQL-Tutor offers six levels of feedback to the student each level determining how much information is provided to the student (see Figure 5.1). These six levels of feedback are positive/negative feedback, error flag, hint, all errors, partial solution and complete solution. Regardless of the level, the student is always given the total number of errors resulting from their submission. Of the six levels of feedback, *positive/negative* is at the lowest level providing the student with only verification knowledge, that is, whether the solution is correct or incorrect. If the student makes an error in an SQL clause, this is indicated by an *error flag* warning which tells the student the clause where the error has occurred. The student can also receive an *hint* message which is level 3 feedback. The hint message provides more information about the type of error. It achieves this at two levels. At first the content of the hint message is topic specific, reciting the general principles of the domain that have been violated. In the second instance, the hint message is more response specific, pointing the user more directly to the error and how it can be corrected. The *all errors* message type provides hint messages for all student errors while *partial solution* feedback displays the correct content of the clause in question. Finally, the *complete solution* displays the pre-specified ideal solution corresponding to the current problem.

Problem Selection SQL-Tutor provides a selection of the various SQL clauses and a wide range of problems to be tested. While the user is able to select the clause and problem to be tested, the system also makes suggestions based on the progress of the student as judged by the student model. The student can work through the system generated order of problems by simply selecting the systems choice option or using the menu of problems to select their own choice. This feature provides the flexibility of allowing students to go backward and forwards through the spectrum of problems as

they require. Students can go through problems in a pre-specified order as they appear numbered by clicking the *next* option.

When the system choice is used, the ITS examines the student model and selects the next problem as follows: First, it uses the number of violations of each constraint to identify a constraint that the student has not yet learned. It then finds a problem whose solution is modeled by that constraint, and presents that problem to the student. In so doing the system is able to choose a problem that exercises the concept which generated the initial error and which can support and test further the student's acquisition of that particular concept or skill. If the student has been selecting problems themselves, then the system can also identify constraints that have not been relevant to any of these selections, and select a problem for which that constraint is applicable. Allowing the system to select problems gives a good coverage of all concepts which the student should possess. Student-determined problem selection on the other hand can have unpredictable effects on the coverage of domain concepts but would be suitable for students who know exactly the areas where they lack knowledge.

2.3.2 Student modelling in SQL-Tutor

The student model used in SQL-Tutor is the constraint-based model which represents knowledge about the SQL domain as a set of constraints on correct solutions. The constraints partition the universe of possible solutions into the correct and the incorrect ones [Mitrovic and Ohlsson 1999]. Each constraint represents a domain concept which cannot be violated if the student is to produce a correct solution. See [2.1.2] for more on constraint-based modeling as previously discussed.

The very first constraints used in SQL-Tutor were formulated by Dr. Antonija Mitrovic [Mitrovic and Ohlsson 1999]. These were developed using two main sources of information: analysis of the target domain and comparative analysis of correct and incorrect student solutions recorded while. Recall that a state constraint is an ordered pair (C_r, C_s), where C_r is the relevance condition and Cs is the satisfaction condition.

In SQL-Tutor, relevance and satisfaction conditions are patterns that match parts of the students solution or the ideal solution, while others are Lisp predicates. Relevance and satisfaction patterns can be arbitrary logical formulas, containing any number of atomic predicates. Each constraint is associated with a number which allows the system to uniquely identify each constraint. Constraints also contain a field which holds a natural language description of the domain state the constraint represents. This can be used in providing error-specific feedback to the student. Finally each constraint specifies the name of the clause to which the constraint applies.

The system distinguishes between two types of constraints, syntactic and semantics constraints. Syntactic constraints refer only to the students solution and represent syntactic properties of queries e.g. use of incorrect command keywords or incorrect command formats. Semantic constraints operate on the relation between the students solution and the ideal solution and represent the semantic properties of queries. Constraint 3 (see Code 1), the number 3 uniquely identifying that particular constraint, is an example of a syntactic constraint, a constraint on the form of a query. It specifies that part of the SQL domain which requires that the FROM clause of a SQL query never be empty. This constraints is always relevant and its relevance condition reduces to "t", the Lisp symbol for a condition that is always satisfied. The satisfaction condition of the constraint checks that the FROM clause of the student submitted query is not null, that is, the FROM clause of the student solution (represented by the variable *ss*) is mandatory. This is a straightforward combination of atomic Lisp predicates. The second field of the constraint is the textual description which forms part of the instructional feedback message given to the student when the constraint is violated. The last part of the constraint is the name of the clause (the FROM clause in this case) and is included in any instructional message [Mitrovic and Ohlsson 1999].

Constraint 360 (see Code 2) is an example of a semantic constraint. Such constraints test the intended result the query is meant to have as interpreted from the query statement. They test whether the commands and symbols used by the student achieve what is required by the query and quite often involve some comparison between student

Code 1 Example of an SQL-Tutor Syntactic Constraint

```
'(p 3
   "The FROM clause is a mandatory one.
    Specify the tables to retrieve data from."
   t
   (not (null (from-clause ss)))
    "FROM")
```

solution and the ideal solution (*is*). Constraint 360 tests that the student has specified ascending order (ASC) rather than descending order (DESC) in the ORDER BY clause as this is what is required by the query statement.

Code 2 Example of an SQL-Tutor Semantic Constraint

```
'(p 360
   "You should have ascending order in the ORDER BY clause,
    not descending!"
    (and (not (null (order-by is)))
         (not (null (order-by ss)))
         (null (intersection'("ASC" "DESC") (order-by is) :test 'equalp)))
    (null (member "DESC" (order-by ss) :test 'equalp))
    "ORDER BY")
```

2.4 Feedback in ITS

Today's computer-based instruction systems provide a self-contained but adaptive learning environment for the user. Designers of such systems need to pay especially close attention to the kinds of feedback they incorporate but guidance in this regard can be vague and vary from source to source. This is because research in traditional learning settings, while clearly demonstrating feedback's importance, has not shown any particular kind of feedback to be universally superior. While much work has been done on the many existing feedback forms, the findings of the effect of such forms on learning have been rather inconsistent and often conflicting (see reviews by [Azevedo and Bernard

1995*a*] [Mason and Bruning 2001]). As such multiple types and forms of feedback have been investigated throughout a wide variety of domains and instructional contexts. One way in which feedback has classically been categorized is based on outcome feedback types such as knowledge of result or response (KR), which indicates whether a response is correct or incorrect, knowledge of correct response (KCR) which provides the correct response or solution to a given step or task, and answer until correct (AUC), which provides both KR and also prevents the student from moving on before the item has been correctly answered [Kulhavy and Stock 1989] [Mason and Bruning 2001]. Feedback messages can also be classed into three categories, informational, topic-specific and response-specific. Informational feedback focuses on declarative knowledge and does not specifically address individual responses but rather provides information relevant to a given problem state such that the correct answer can be drawn. In contrast topic-specific feedback gives more specific information about the target question or topic leading the learner through the correct answer. It does not however address correct or incorrect responses. Of these, the most successful is response-specific which provides feedback on the correct answer and incorrect steps. In cases where an incorrect action has been chosen response-specific feedback explains why the selected action is incorrect and hints the student towards the correct answer.

At the simplest level though feedback is classified as positive or negative feedback. Simply put, positive feedback is feedback provided based on correct responses whereas negative feedback is in response to incorrect actions. In this research we consider positive feedback and how it can be used to improve the learning experience in ITSs.

2.4.1 Factors Influencing Effective Feedback in Learning

As important as feedback is to learning, so is the design of such feedback, the presentation, scheduling and content. The content of the message consists of a verification component which indicates the performance level and action outcome (e.g. correct) and the informational component containing declarative information on the task, error(s),

correct step or solution(s). Several branches of research have attempted to identify the factors which contribute to effective feedback. Nkambou and Kabanza [Nkambou and Kabanza 2001] identify three general factors as:

- the nature and quality of feedback messages
- the characteristics of the instructional context
- individual characteristics of the learner

They go further to state that the nature and quality of feedback is determined by instructional goals and objectives, technical and formal issues related to presentation of feedback messages, and the semantics of the feedback message. Other researchers have identified several key dimensions that may influence the effectiveness of feedback. These include elaboration, student achievement levels, depth of understanding, attitude toward feedback, learner control, response certitude, and timing. Finally issues such as learning tasks, sources of problems, errors, goals and physical parameters of the environment characterize and influence the instructional context under which feedback is given.

2.4.2 Presentation of Feedback

The way in which feedback is presented has the potential to inhibit or enhance a user's overall performance. This includes also the order of feedback presentation, the type of feedback presented (e.g. knowledge-of-correct response (KCR) or response-contingent), as well as the communication mediums used to provide the feedback. Some studies for example [Narciss and Huth 2004] suggest that no feedback should be presented before learners have attempted to solve the learning task, especially KCR feedback. This is because there is a high risk that the student may not make enough of an effort to solve the problem if feedback is presented prematurely before the student can process the problem or solution. To reduce cognitive load it is also suggested that

elaborated feedback information be presented stepwise in manageable pieces so that the learner can adequately treat and process the information presented.

As previously mentioned, the communication module in intelligent tutoring systems contains both the interface and optionally, a stored representation of the communication knowledge including positive and negative feedback. The interface provides the platform for the interaction between the student and the system providing the graphic user interface (GUI) by which students communicate with the system. Interface design is influenced by multimedia learning theory which deals with using the potential of multimedia systems to facilitate smooth, easy and more efficient interaction between users and systems and thus increase learning. The theory [Mayer 2001] has yielded several design principles which can be used (see Section 2.1.1 for more information).

2.4.3 Content of Feedback Messages

In the meta-analysis of research on the effectiveness of feedback in computerized instruction involving 22 computer-based training studies [Azevedo and Bernard 1995*b*] Azevedo and Bernard found strong evidence that students given feedback (immediate and delayed) performed consistently better than students without feedback. Additionally, a number of studies have found that elaborated feedback, in which students are helped to find the right path, is more effective than situations in which they are simply told whether they are right or wrong.

Early CBI research by Gilman [Gilman 1969] compared student learning after receiving one of five types of feedback: no-feedback, knowledge-of-response, knowledge-of-correct-response, response-contingent, and a combination of knowledge-of-correct-response and response-contingent feedback. Knowledge-of-response (KR) and Knowledge-of-correct-response (KCR) have already been explained. Response-contingent feedback commonly referred to as extra-instructional feedback, provides not only verification information but also submission-specific elaboration response. It gives response-specific feedback that explains why the incorrect answer was wrong and also

why the correct answer is correct. Results of the study indicated that simple knowledge-of-response feedback did not improve learning above the no-feedback condition, but that knowledge-of-correct-response, response-contingent, and a combination of these types of feedback can significantly improve student understanding.

Similarly, research by [Whyte, Karolick, Neilsen, Elder and Hawley 1995] revealed that the higher the levels of feedback, the higher the learning gains in response to computer based instruction (CBI). More specifically his results revealed that students who used CBI giving knowledge-of-correct-response feedback along with additional information (previously referred to as response-contingent) scored significantly higher on concept acquisition tests compared to students receiving other forms of feedback (knowledge-of-response, knowledge-of-response plus additional information, and knowledge-of-correct-response). While many forms of feedback exist, there must be some ideal form. [K. Zakharov 2005] says that "an effective feedback message should tell the user (a) where exactly the error is, (b) what constitutes the error (perform blame allocation), (c) refer the user to the underlying concept of a correct solution (revise underlying knowledge) and (d) possibly tell the user what has to be done to correct the error. This statement is based on the Ohlson's theory of learning from performance errors [Ohlsson 1996].

These studies and many more show clear benefits for feedback involving elaboration information. It seems that the extra information provided through the use of elaboration feedback allows students to correct conceptual errors whereas simple response regarding the correctness of the answer, in contrast, had no significant impact on learning rates and outcomes.

2.4.4 Scheduling Feedback

Research on the timing of feedback has a long history dating back to the 1920s. It is a history that that is stained with conflicting results and interpretations, and failures due to replication. One of the more controversial and intensely researched areas of feedback

scheduling has been on the effects of delayed vs immediate feedback. [Azevedo and Bernard 1995*b*] states that providing feedback immediately provides the best instructional advantage for students. [Kulik and Kulik 1988] concluded "that delayed feedback appears to help learning only in special experimental situations and that, more typically, to delay feedback is to hinder learning". It showed specifically that immediate feedback is more effective than delayed in applied studies and list-learning, but is less effective for acquisition of test content. Other researchers [Schooler and Anderson 1990] suggest delayed feedback to be better. They suggest that it fosters the development of secondary skills such as self-correction and error detection because unlike immediate feedback, it does not compete for working memory resources forcing out information necessary for operator compilation. The study revealed a general advantage for delayed feedback in terms of errors, time on task and percentage of self-corrected errors. Many intelligent tutors out there today implement both immediate and delayed level feedback.

SQL-Tutor for example [Mitrovic et al. 2001] delays giving feedback and hints on steps until the student has indicated that the solution is complete or the student prematurely makes a submission to investigate the correctness of a step. It does not allow the student to evaluate a step or receive any type of feedback without evaluation of the overall solution; this allows the student to practice finding their own errors and correcting them. Other ITSs give feedback on a step as soon as it is attempted and claim to prevent long, unproductive trial-and-error searches and are best suited and intended for novices who would still be struggling to find any solution at all [Anderson et al. 1995][Anderson and Corbett 1994]. Other systems however have moved towards a merger of both immediate and delayed feedback being utilized within a single instructional system. It is thought that this hybrid approach could bring the benefits of both feedback types and facilitate more effective student learning as students can have immediate knowledge of the correctness of their response, but still have time to think about the error before informational feedback is given.

Scheduling of feedback also encompasses the issue of the frequency with which feedback should be given. It has been shown that low working memory load are closely

related facets of well-practiced performances. For example automatized tasks require less memory capacity for the very same reason that you are not required to think about the task being performed in order to do it. As a learner becomes better at performing a task, he or she begins to feel less strain when performing the task and cognitive load is reduced. If feedback is presented frequently [Schooler and Anderson 1990], studies have shown that it increases working memory load as cognitive processes begin to compete for finite working memory resources. This can be a significant impediment to learning and must be considered within the context of the domain when designing feedback in intelligent tutoring systems.

CHAPTER 3

Research Goals and Hypotheses

3.1 Goals

This research aimed to fulfil several specific objectives. These were to:

- Study and understand positive feedback within various cognitive theories of learning such as the ACT theory of learning and problem solving [Anderson 1983] and Ohlsson's theory on Learning from performance errors [Ohlsson 1996] .
- Determine what pedagogical content knowledge and strategies are used in exercising positive feedback in and tutoring in general and more specifically in the tutoring of SQL queries. Specifically identify and map how experienced tutors are able to identify and utilize the technique of positive feedback during tutoring.
- Use the understanding of these theories and concepts of positive feedback to the development a systematic approach to positive feedback
- Design and implement this systematic approach to positive feedback in SQL-Tutor by extending SQL-Tutor to give positive feedback

- Evaluate the effectiveness of the systematic use of positive feedback as supported by our extended version of SQL-Tutor

3.2 Hypotheses

In line with our goals, our hypotheses were as follows:

- That many student steps are tentative; the student is guessing, or at least rather uncertain as to what to do. Given the pressures to do something, they do, but don't really know ahead of time that it is the right thing to do,
- that under conditions of uncertainty if the student happens to be correct, providing feedback helps the student to be less uncertain about what to do in that type of situation,
- that positive feedback works to reinforce existing knowledge and in some cases create new knowledge and by so doing reduces both the time taken by students to solve problems and the number of errors made,
- that the constraint base underlying student modeling in constraint-based tutors can be used to predict uncertainty in student actions, and
- that at least within the short-term learning period, the learning rates would be higher in a positive-negative feedback ITS compared to a negative feedback only ITS.

3.3 Positive Feedback in Learning

Current ITSs focus on negative feedback or corrective feedback as it is sometimes called, in which information is provided by a teacher or other instructional agent to correct the errors a learner has committed. This approach to Intelligent Tutoring Systems is in line with cognitive learning theory which continues to stress the error-correcting

mechanism, including the ACT-R cognitive architecture, a theory for simulating and understanding human cognition and acquisition of cognitive skills [Anderson 1983]. Although they accomplish it in quite different ways, both the model tracing approach and the constraint-based modeling approach [Ohlsson 1993] work by identifying faulty knowledge and correcting such knowledge through feedback. Negative feedback decreases the chance of committing the same error in future situations where the same error might occur.

"Cognitive learning theory as a whole however also includes the principle that people also learn from positive feedback. It is, in fact, the combination of error correction and strengthening that gives the power law learning curve, not the error correction mechanisms by itself" [Ohlsson 2006]. ACT-R theory of skill acquisition strongly tied to the tutoring effect states that the application of weak knowledge can result in slips and errors. Apart from practice which works to produce smoother, more rapid and less errorful execution of tasks, positive feedback has also been noticed to improve the strengthening effect in tutoring.

Consider the scenario where a student is attempting a problem but is not entirely sure of what to do. That student uncertain about what to do, makes a tentative attempt at the next step and surprisingly it turns out to be correct. The student is likely to store that piece of 'correct' knowledge if they are aware that their action is indeed correct. There must therefore be some feedback mechanism which confirms to the student that their action is correct. Experienced human tutors support this type of learning process by providing feedback to confirm the correctness of the students action. This is only one of the many situations where positive feedback is used quite effectively to support student learning. It is present in many disciplines including psychology and computer science. Self-explanation is one area of psychology where this is observed. A student is more likely to store a concept if he/she can successfully explain that concept. Another example which occurs in computer science is explanation-based learning of correctness implemented in Cascade [Vanlehn, Jones and Chi 1992] a computational architecture that models human learning and problem solving. When studying an example, Cascade

must self-explain each worked step. When the system finds a step it cannot explain, this signifies a knowledge gap. Because the example specifies what the answer is, Cascade can (given appropriate domain-general background knowledge) compensate for the gap by constructing an explanation for the answer. When Cascade successfully creates such an explanation, it learns a new piece of knowledge that it has some faith will work in future problems.

It therefore appears that one of the ways in which positive feedback works is by reducing the number of tentative steps made by a student, creating and storing new knowledge chunks in cases where the student was lacking, or strengthening in situations where there was uncertainty and doubt. This research considers these issues in more depth.

CHAPTER 4

Systematic Approach to Positive Feedback

In this research we investigate the reasons why positive feedback works. Research into the behavior and methods used by expert tutors suggest that experienced tutors use positive feedback quite extensively and successfully. If current ITSs are to improve in performance and quality of learning experience delivered, they must also include this widely used and successful form of feedback.

It can be difficult and is quite often puzzling why positive feedback works. You are providing information to the student, sometimes containing no information or knowledge of the domain. This information is often short, very simplistic and often is of no relevance to the material being delivered. Furthermore, such feedback by definition follows the correct actions of the student and as such would seem of nil effect, coming after the fact. Data however shows the experienced tutors do use it and use it quite effectively. We therefore ask ourselves why positive feedback works. What function does positive feedback have and where does it fall within learning theory?

Ohlsson proposes the following hypothesis which we will investigate within this research project. The hypothesis suggests that most student steps are tentative meaning

that the student is guessing or is at least uncertain as to what to do. Positive feedback therefore works by helping to remove/reduce the amount of uncertainty associated with student actions. Students are pressured to provide answers, to come up with solutions either constructively or to simply make it up. In either case, if such a move happens to be correct then providing assurance to the student of its accuracy helps to reduce uncertainty and apprehension.

4.1 Scheduling Positive Feedback

4.1.1 Timing of Positive Feedback

In general, tutoring systems vary on their policies about when to give feedback. Scheduling policies typically fall under three broad categories, immediate feedback, delayed feedback and demand feedback. Immediate feedback as observed in ANDES, Algebra Cognitive Tutor [Gertner and VanLehn 2000] and AutoTutor [A.C. Graesser and TRG 2001], give feedback immediately after every student step. Sherlock [Katz, Lesgold, Peters, Eggan and Gordin 1998] gives feedback mostly after the student has finished troubleshooting, that is delayed feedback and immediate feedback only if the student step violates a safety regulation. In SQL-Tutor feedback is given only when the student clicks on the Submit button, in other words when the student demands feedback. More complex policies are possible where the policy is a function of the student's competence. Consider for example a student who is approaching mastery of a given domain. It would make sense to provide less feedback to such a student effectively delaying the feedback as compared to a less competent student who might need a more immediate feedback and prompt feedback intervention scheme.

The above feedback policies have been considered in formulating the scheduling of positive feedback. In one-to-one human tutoring, the student typically makes an attempt at the problem before the tutor has any intervention, at which time it is usually based on some error that the tutor has identified in the student's solution. The student continues

to make such attempts until the problem is solved, effectively submitting each attempt for the attention of the tutor who provides feedback. We propose that positive feedback be given on a demand basis as is currently facilitated by SQL-Tutor and to mimic the actions of human tutoring. However, the system should not give positive feedback on each correctly used constraint, as the amount of such feedback would be overwhelming. Instead, we identified events in the student's tutoring experience which should trigger positive feedback. It is at these points that we hypothesize positive feedback is most effective. The positive feedback provided to a particular student will depend not only on the submitted solution, but also on the student's knowledge (as captured by the student model) and the state of interaction. We developed four general cases when positive feedback should be given:

- When the student is expressing uncertainty but nevertheless does the right thing
- When the student is too paralyzed to do anything at all and requests assistance.
- When the student has overcome aspects of the domain commonly agreed upon as being difficult and challenging
- When the problem or task has been successfully completed and at other major goals within the tutoring session

Giving student feedback under uncertainty

In this section we consider giving positive feedback under uncertainty that is, when the student is expressing uncertainty but nevertheless does the right thing. Consider a situation where a student is uncertain or simply does not know the correct format for the date within SQL. If the student gets the format wrong, the first time or a number of times after that, then we are certain of the student's knowledge of SQL date formats. It is either they are uncertain of the format, maybe toggling between two options or simply do not know and is guessing.

In either case the student has submitted an incorrect answer and based on our hypothesis, if the student is uncertain, that is, choosing from several chunks of knowledge or problem strategies trying to find the correct one or simply guessing (scientific guessing), positive feedback will help reinforce those knowledge chunks which led to the student obtaining the correct answer. In one-on-one human tutoring uncertainty is typically dealt with through probing and providing hints. Tutors typically use scaffolding techniques to break the problem into simpler components isolating the knowledge component where the uncertainty occurs, encouraging the student to make an attempt and providing hints and suggestions until the student has mastered the concept. Correct responses during this period of learning are often followed by a series positive feedback messages geared at reinforcing these newly learnt or successfully applied domain principles.

Giving positive feedback under paralysis

During tutoring there will be moments where the student is paralyzed, that is, totally and absolutely unsure of the next step. In such situations the student is unable to proceed without additional aid and feedback comes into play. Sometimes however low-level negative feedback is not sufficient, particularly when the student is lacking some incite knowledge critical to solving the problem. Therefore despite tutorial negative feedback the student does not obtain the correct solution and even when negative feedback is given, the information provided does not seem to assist the student in any way and the student is no better off than previous. Continuous attempts at obtaining correct solutions result in violated domain principles.

In other cases the student might be oblivious as to the next step or to the solution path having no clue or inclination as to what is a correct strategy to solve the problem. In this case the student is too paralyzed to do anything and can only move forward through the assistance of the tutor in the form of some more direct hint on what the solution should be.

In either case, if more specific but indirect feedback is given, that is, the student is given an hint to the answer but not the answer itself, and the student successfully uses this added knowledge to correctly infer the answer, then positive feedback should be given. It shows that the student had at least some knowledge of the problem requirements which they successfully applied. Based on our hypothesis positive feedback should increase student confidence. It will also decrease uncertainty over the student's existing knowledge, decrease uncertainty over the knowledge given within the hint statement which was once lacking and strengthen the connection between these two pieces of knowledge chunks.

Giving positive feedback under certainty

There will be instances when the student is absolutely certain of their actions whether as a result of prior knowledge or learnt through the system via error correction in prior problem steps. We have identified two instances under certainty when feedback should be given.

Giving positive feedback at difficult and challenging domain concepts Certain parts of the domain are difficult and challenging increasing cognitive load and the amount of active information processing required from the student. This requires not only domain knowledge (e.g. facts, concepts, events, rules, theories and models) and task specific procedural knowledge (e.g. sorting and drawing) but also the development of more specific and general meta-cognitive skills. In this research we propose giving positive feedback when the student correctly and adequately deals with such situations. Of course this will vary from domain to domain. This will require cognitive task analysis to identify domain specific knowledge, the cognitive skills (e.g. classification, inference and memory) related to these items and the expert skills required to master these tasks.

Giving positive feedback at major goals within the tutoring session When the student gets the entire problem correct then that represents a significant achievement. If

a student for example, gets a very difficult and complex problem correct on their first attempt or satisfies a difficult constraint the first time it is relevant, then it potentially signifies that the student has mastered those aspects of the domain addressed by that particular problem or constraint. In this case we propose giving positive feedback to indicate to the student the magnitude of their accomplishment and to reinforce that correct response.

4.1.2 Frequency of Positive Feedback

The question of when feedback should be given leads unavoidably to the question of how often feedback should be given. In [Kluger and Denisi 1996] it was argued that for over a century feedback interventions (FIs) were producing negative but largely ignored effects on student performance. The authors in the paper proposed a preliminary FI theory or FIT based on the central assumption that FIs somehow changed the locus of attention among three general and hierarchically organized levels of control. These were from bottom to top:

- The actual learning of the task
- Motivation to complete and perform the task
- Meta-tasks processes

Results of their moderator analysis suggested that effectiveness of providing feedback decreased as the student's attention moved up the hierarchy closer to the self and away from the task. Anderson and Schooler in [Schooler and Anderson 1990] consider the disruptive potential of immediate feedback. Results of their experiments suggested a general advantage for delayed feedback with regards number of errors made, time spent on task and the percentage of errors that subjects were able to self-correct. It was suggested that this was because immediate feedback increased competition for working memory resources which would otherwise be used to facilitate task specific procedures.

Because of the negative consequences and side effects of providing feedback too often, it is important to determine the frequency with which positive feedback should be given. This is particularly crucial because the information being provided with positive feedback is based on correct student responses and depending on the level of the student, has potentially less useful information than negative feedback messages. For example, giving positive feedback in instances when the student is absolutely certain of the answer should be avoided, especially where the student has mastered a given concept. This could potentially worsen the negative impacts of current and future feedback interventions, both positive and negative.

Along with the scheduling previously proposed for positive feedback, we also propose regulating the frequency with which positive feedback messages is given. When positive feedback is given under student uncertainty it is important to determine what the source of the error is and to not continually give feedback for the same error. We propose that once positive feedback has been given for a particular domain concept, further feedback should not be given unless the student has had at least five (5) more attempts involving that domain concept. This will avoid giving feedback too often, particularly where the student has only made a slip (an occasional action which is not systematic and which the student can correct).

4.2 Selection and Specification of the Content of Positive Feedback

Feedback from a cognitive standpoint is used to provide the learner with information required to foster procedural and declarative knowledge development and development of metacognitive skills through verification and elaboration. The feedback message and its usefulness is determined by a number of factors including the instructional context and technical aspects such as timing and frequency of messages as eluded to earlier. The nature and quality of the feedback message however also depends quite expectedly

on the content of the feedback message. When treated as a unitary variable, the form or composition of feedback can be viewed as ranging along a continuum from a simplistic "Yes-No" format to extending the content of the response to provide substantial remedial or corrective information in some cases, adding new but related knowledge. Movement along this continuum inevitably leads to more complex feedback content and effectively forming new instruction. It was this type of insight which lead away from the behaviorist view of feedback solely in response to correct actions to feedback in response to incorrect responses[Kulhavy and Stock 1989].

One such insight which lead to the development of the CBM approach was Ohlssons theory of learning from performance errors, a theory which states that incorrect student actions point to the existence of faulty knowledge structures within the students perception. Based on this theory, learning is broken down into the two separate and distinct parts: error detection and error correction. The theory states that in order for the learner to detect an error, the action performed must result in the production of some perceptible effect either auditory, visual or stimuli, which manifests itself within the task or learning environment. The learner must then be able to observe and interpret this effect as a signal of an error and while in some situations, this process can be trivial such as shooting basketball hoop, it can be quite difficult in others where the error signal can be hidden, indirect or significantly delayed. Once the error has been detected, the learner can then begin to infer the cause of the error either from the erroneous effect itself or using some declarative knowledge. Correcting an error might in the simplest case be a one step fix but quite often it can be a process of trial and error depending on the given task.

In constraint-based (CBM) tutors, constraints are used to implicitly represent all possible erroneous solutions in the knowledge domain, and so the burden of error detection and blame assignment is taken away from the student and carried out by the system. This is not to say that the student will not detect errors of their own, but where they fail to, the system will pick it up. When an error has been detected, the ITS should then point the student to the source of the error, usually in the form of a feedback message.

[K. Zakharov 2005] says that "an effective feedback message should tell the user (a) where exactly the error is, (b) what constitutes the error (perform blame allocation), (c) refer the user to the underlying concept of a correct solution (revise underlying knowledge) and (d) possibly tell the user what has to be done to correct the error." According to [Kulhavy and Stock 1989] feedback has one of two effects on each response that a learner makes: (a) letting him know when a response is right or wrong and (b) to correct or facilitate correction when a wrong response is given. These two types of information are commonly referred to as verification and elaboration, verification being the simple judgement of an answer being correct or incorrect and elaboration being information specific to the domain or task which guides the learner towards the correct answer. These two approaches can be married, verification information stating that there is an error and possibly the location of error, and elaboration information performing blame assignment, referring the user to the underlying concept and possibly telling the learner what has to be done to correct the error. While most researchers now hold the view that effective feedback must include both type of information, current approaches only provide extensive elaboration under incorrect responses and verification under correct responses, that is, response-specific elaboration. The content of positive feedback when given under these circumstances is usually limited to messages such as "Thats correct" or "Good job! That is the correct answer".

In the design of positive feedback content, we propose that verification information be given as is consistent with prior research and generally accepted. Assuming that the learner is not guessing, this serves to indicate to the him that his interpretation or understanding of the domain concept is correct. More generally it indicates to the student that he is achieving the task goal and that his chosen strategy is being successful. It also indicates the performance level achieved such as complexity level of task performed, percentage of correct answers or distance to learning criterion. We also include topic specific and response-specific elaboration information about the students correct response. If the learner has correctly applied some chunk of knowledge then response-specific information works to confirm the learners specific actions ap-

plied while topic-specific information provides general and possibly new information related to the correct response, not only confirming the learners knowledge but possibly providing and creating new knowledge. Creating this content will involve the use of techniques of cognitive task, error and correct response analysis.

The results of task, error and correct response analyzes will provide information needed to select those general and task-specific informative components matching the task requirement. We propose an approach adopted and adapted from [Narciss and Huth 2004] which can be used to create elaborated information content for positive feedback messages (see Figure 4.1).

4.3 Selection and Specification of the Form or Modus of Positive Feedback Presentation

While selecting and offering the right and appropriate content information to the learner is a necessary condition for effective feedback, it is by no means a sufficient condition. The way in which the learner processes feedback is also strongly influenced by and can in fact be inhibited by the form or type and method of feedback presentation. We have in previous sections mentioned the importance of the communication module which contains both the interface and optionally, a stored representation of the communication knowledge. It is charged with controlling the very important and learner sensitive interactions between the learner and the system. The interface must present clearly and concisely, with as little as possible use of human memory load, feedback to the student.

A very important question which requires answering is whether positive feedback should be distinguished from negative feedback in presentation. In [Eugenio, Lu, Kershaw, Corrigan-Halpern and Ohlsson 2005] the authors implemented an ITS teaching students to extrapolate complex letter patterns. In the neutral version of the system students receive green for a correct answer and red for incorrect answer while in the positive version, students received feedback via the same color coding scheme as well

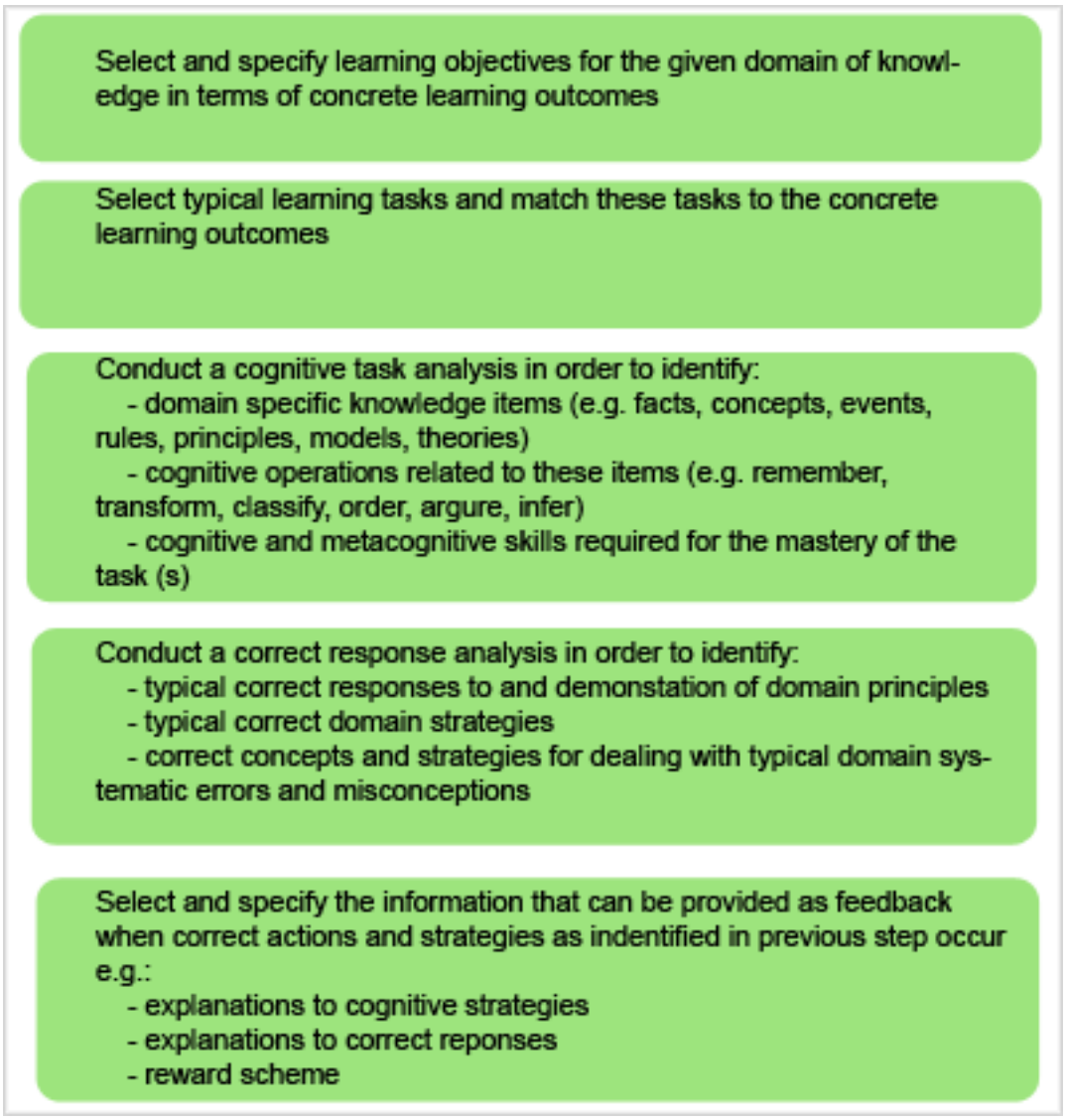

Figure 4.1: Designing Positive Feedback Content Using Cognitive Task and Correct-response Analysis

as verbal feedback on correct responses only. The overall results of this study proved to be inconclusive and no real distinction was found in using verbal feedback, however it must be noted subjects in the positive condition seemed to do slightly better than subjects in the negative condition although those subjects made fewer errors. The importance of this study for purposes of this research however is that they experimented using several variations in giving simultaneously, positive and negative feedback. Aside

from the above study, most of the work already done in providing both positive and negative feedback seem to present both types of feedback in the same form and using the same methods e.g. text. Admittedly the positive feedback provided in these tutors have been limited to verification information lacking elaboration of any type.

In general we propose that positive feedback be presented in the same manner as negative feedback, that is, all feedback should be presented in the same form, using the same human input channels to reduce the cognitive load of the learner having to decipher two distinct forms of presentation.

To avoid and reduce cognitive overload due to mode of presentation, we propose that positive feedback be presented in line with the cognitive theory of multimedia learning [Mayer 2001]. This theory outlines how technology can be used alongside cognitive psychology to foster learning. The theory is influenced by the dual coding theory first advanced by Allan Paivio [Paivio 1986] which states that the learner possesses both a visual and verbal information processing system the former responsible for processing information retrieved via visual stimuli and the later for information received via auditory stimuli. [Moreno and Mayer 1997] presents five major principles which should be adhered to when preparing positive feedback and feedback in general:

- Multiple representation principle: it is better to present an explanation in words and pictures than solely in words
- Contiguity principle: when giving a multimedia explanation, present words and pictures contiguously rather than separately
- Split-attention principle: when giving a multimedia explanation, present words as auditory narration rather than as visual onscreen text
- Individual Differences Principle: The foregoing principles are more important for low-knowledge than high-knowledge learners, and for high-spatial rather than low-spatial learners.

- Coherence Principle: When giving a multimedia explanation, use few rather than many extraneous words and pictures.

We propose therefore that positive feedback particularly complex feedback messages be presented based on the above design principles using acoustic presentation where possible to reduce single channel overload.

CHAPTER 5

Design and Implementation

5.1 The Choice of Domain and ITS

The domain chosen to conduct this research needed to be previously researched and well understood and well documented in terms of pedagogical strategies, student behavior and student learning patterns. It was thus decided that a new ITS would not be built but rather an existing system should be modified to include the new components of the system. The domain chosen was SQL, and it was decided that the system to be modified would be SQL-Tutor.

Recall that the main aim of this research was to develop, implement and evaluate a systematic approach to positive feedback in a constraint-based tutor. It was important that a well understood and previously researched domain especially in the area of negative feedback be chosen so that the effects of giving positive feedback could be adequately measured and comparative analysis performed. As a domain, SQL is very much a part of every university syllabus, and knowledge of this language is required of every computer science student developing modern day computer applications. SQL has been taught at the University of Canterbury and was the first domain to be used in the development of constraint-based ITSs. Work on SQL-Tutor began in 1995 but the

first evaluation study was conducted in April 1998 when constraint-based modelling was first tested as a theory in support of learning. Since then SQL and SQL-based intelligent tutoring systems have been evaluated within the Intelligent Computer Tutoring Group (ICTG) of the University of Canterbury eleven (11) times (including this research) providing years of documentation and understanding in this area. Because constraint-based modelling naturally facilitates learning through error detection and correction, most of these studies have focused on negative feedback and SQL-Tutor has only implemented positive feedback minimally. The system therefore presented a great opportunity to study positive feedback and provided a rich historical comparative base on negative feedback.

There were also more logistical reasons for choosing SQL-Tutor as the ITS for evaluation. Firstly, the research required that an evaluation study be performed where sufficient number of students were available and could be separated into both a control and an experimental group such that both groups could use and have access to the tutor for adequate periods under realistic learning conditions. SQL-Tutor is maintained by the ICTG group and used in certain undergraduate courses at the University of Canterbury and as such provided the setting to conduct the evaluation study in a real learning environment. Secondly, building an ITS from scratch, including modelling the domain and creating problems and solutions, requires a great deal of time and expertise in the domain. In addition, a new ITS would not have the required historical research context and would add nothing to the research as the research sought to evaluate a learning mechanism and its design, rather than a specific software implementation. Finally, because SQL-Tutor is the second-to-last tutor used in the course it not only provided ample time for implementation and but because it was also followed by a lab test in SQL it provided an incentive (the lab test) for students to make adequate use of the tutor.

5.2 Changes to SQL-Tutor

We have indicated previously that several versions of SQL-Tutor have been developed and tested since 1995 when work on the tutor first began. The version of the tutor used in this study is the original version previously described in section 2.3. In this section we describe more precisely the system used and the changes made to implement the experimental version.

5.2.1 Interface

Only one change was made to the interface and this was done for both the control and experimental groups (see Figure 5.1). This was the removal of the hint item from the drop down menu where it was located in the original version and the addition of a hint button to the main task environment of both the experimental and control versions. This was done to facilitate case 3 of the positive feedback scheduling mechanism. Case 3 represents the situation where a student is too paralyzed to take action meaning the student is at a point in the task where he or she is totally and absolutely unsure of what is required and at this point the student requests a hint. In the original version, students can request a hint message by selecting it from a drop down menu where all the various feedback levels are listed, however this option is not visible till the menu has been clicked. We wanted students in both versions of the system, particularly the experimental group to be aware of the hint button and have quick and easy access to it when needed. Needing a hint was used as a signal for uncertainty.

5.2.2 Student Modelling and the Student Modeler

Both versions utilized the same version of student modelling previously explained in section 2.3.2. Constraints-based modelling (CBM) was used for short-term student modelling while long-term models contained constraint histories, knowledge scores (for each constraint), and solved problems. The student modeller in both versions function

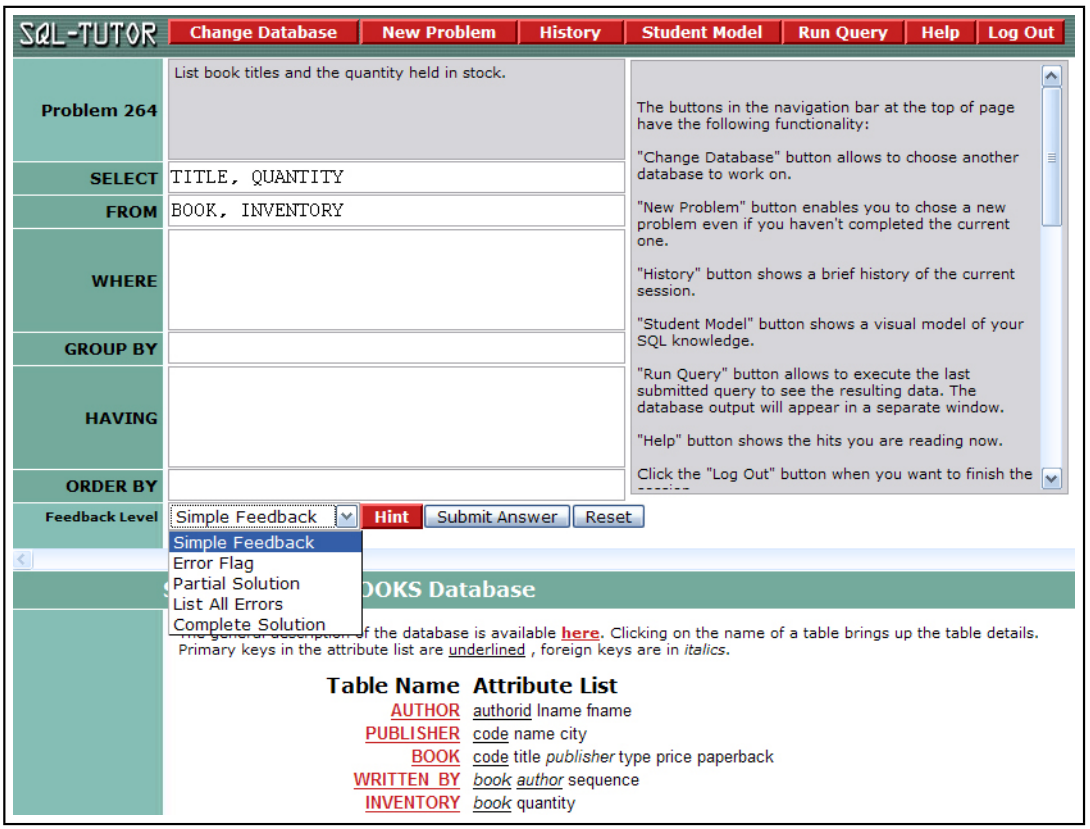

Figure 5.1: The Task Environment in the Experimental Group

as described in section 2.3.1. SQL-Tutor begins by creating two structures, a relevance and a satisfaction network from the compilation of semantic and syntactic constraints. These help to improve the execution speed and efficiency of processing constraints. When checking the correctness of the student's submitted solution, SQL-Tutor compares it to the correct solution that is, the stored ideal solution a two-step process which results in a list of constraint violations.

5.2.3 The Pedagogical Module: Problem Selection

Problem selection in SQL-Tutor like most other ITSs, is controlled by the pedagogical module with input from the student model. Problem selection remains the same in both the control and experimental versions of the system. Two methods of problem selection

are permitted within the system. Firstly the system based on the student model can make suggestions on the SQL clause the student should attempt, then the problem it thinks the student should attempt. Alternatively the student can make both choices for themselves or a combination of student and system choice. The system first suggests the clause the student should attempt which its believes would result in the most learning. At this point the student can override this choice by selecting another clause from a listing of all SQL clauses. The student also has a view of the open model which shows at any given point, the total amount of knowledge obtained in the use of a particular SQL clause represented as a percentage. This provides the student with information about their own knowledge needed to make an informed decision about the clause they should attempt (see Figure 5.2).

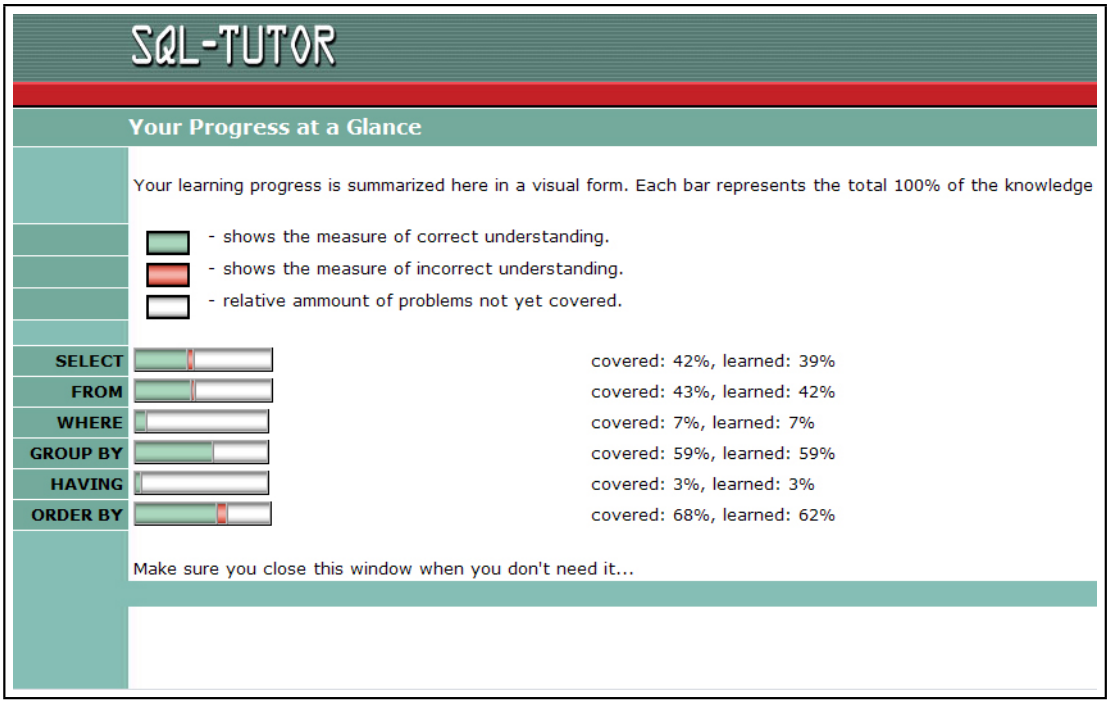

Figure 5.2: The Open Student Model in SQL-Tutor

Once the choice of clause has been made, the student can decide to go with the system choice of problem question or select their own from a listing of all problems related the given SQL clause (see Figure 5.3). It can be argued that for purposes of

analysis, all problem selection should ideally be done using only system choice however this could result in frustration and de-motivation as expertise increases as users would be very constrained. This would also significantly contradict with the whole idea of allowing persons to learn under real learning situations as in the real world individuals typically have some control over what they wish to learn.

When the system choice is used, the ITS examines the student model and selects the next problem as follows: First, it uses the number of violations of each constraint to identify a constraint that the student has not yet learned. It then finds a problem whose solution is modeled by that constraint, and presents that problem to the student. In SQL-Tutor, each problem is assigned a problem difficulty level ranging from one to nine; one being the easiest and nine being the most difficult. This difficulty level is assigned by the problem author and is also used in determining the problem that should be assigned to the user.

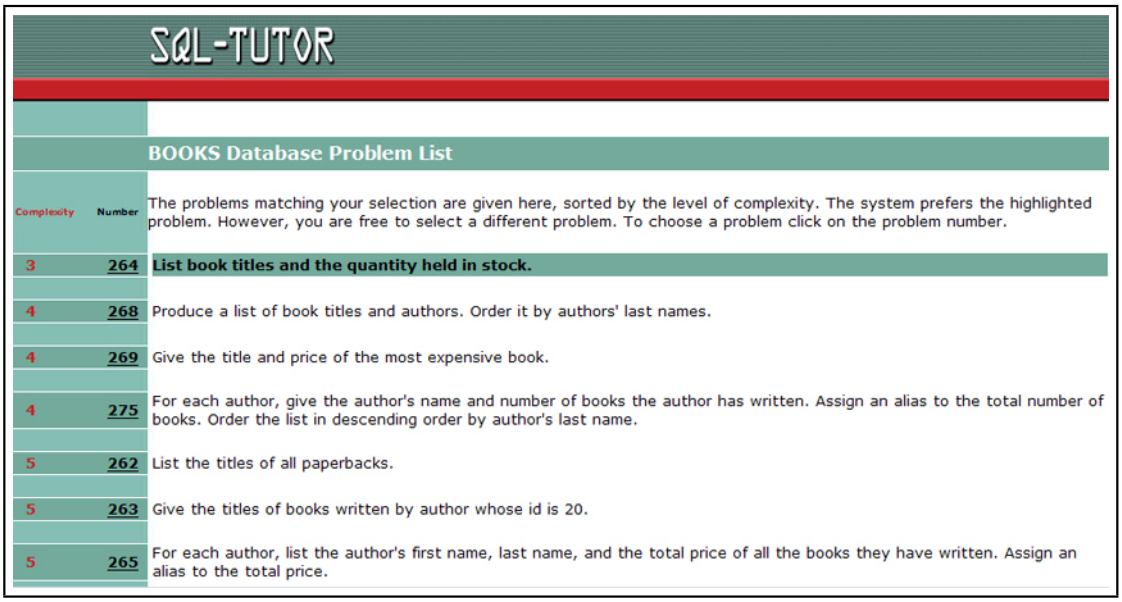

Figure 5.3: The Problem List, with the System's Choice Highlighted

5.2.4 The Pedagogical Module: Feedback

In section 2.3.1 we discussed the generation of feedback messages in SQL-Tutor in some detail. A positive feedback message is similar to the negative one: it points to a part of the solution which is relevant, and explains the domain principle involved. A negative feedback message in addition discusses how the student's solution violates the domain principle. We also discussed the various levels of feedback available including hint level, all errors, partial solution and complete solution. Feedback generation in the control version of the system remained unchanged with only negative feedback being generated from violated constraints as outlined in section 2.3.1. Negative feedback generation in the experimental also remained unchanged. Major changes were however made to the experimental group to implement our approach to positive feedback.

Each of the positive feedback scenarios outlined in Chapter 4 were mapped using the student model and other metrics to specific feedback cases. Each case represents a trigger, such that whenever all the conditions of the given case are satisfied, a positive feedback message is generated and displayed. These cases have been outlined in the section which follows and detailed examples presented. Cases were implemented using a series of functions within the pedagogical module, each function implementing the behavior specified by the given case. These cases are described in greater details in the sections that follow. In the experimental group, both positive and negative feedback were presented simultaneously and when necessary, positive feedback was presented first followed by negative feedback. If a student's solution was incorrect, but still matched one or more of the positive feedback cases, then positive feedback was presented first, followed by negative feedback. Figure 5.4 illustrates a situation in which the student received some positive feedback about join conditions, and then negative feedback, on the Error Flag level, pointing the student to a mistake in the FROM clause. In the sections that follow we present each case in more detail giving examples of each.

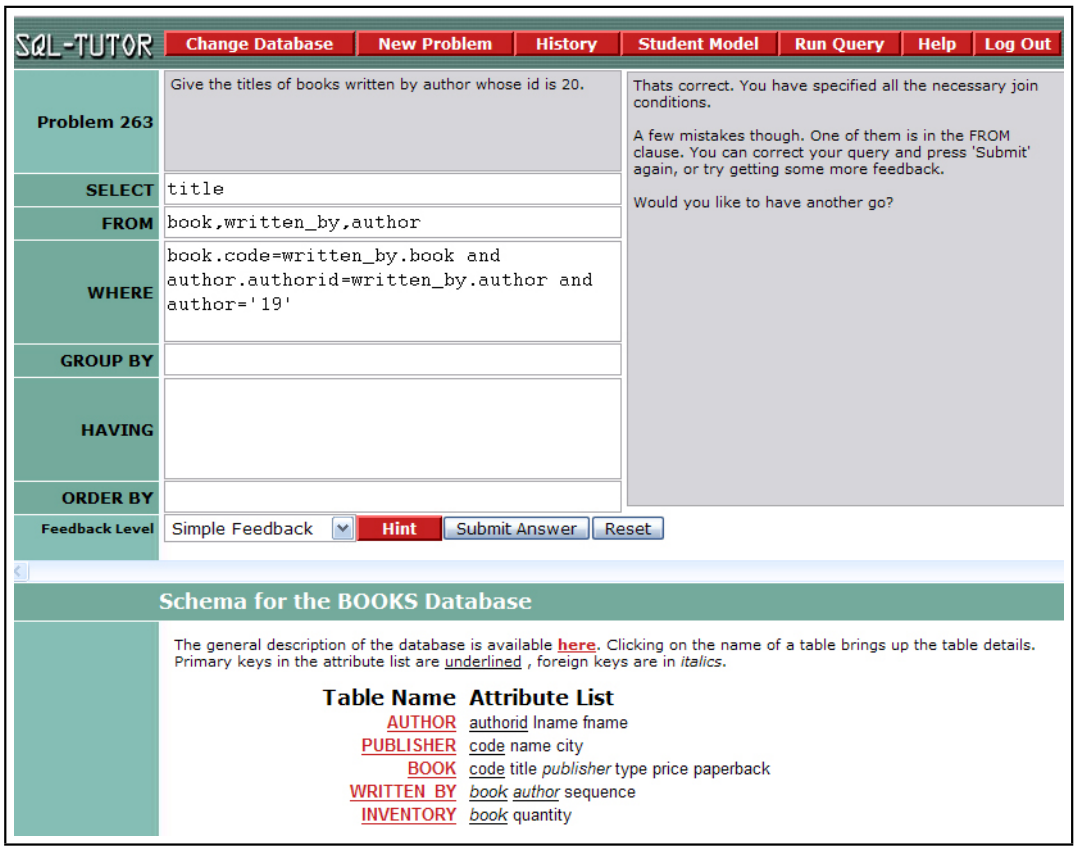

Figure 5.4: Giving Negative and Positive Feedback Simultaneously in SQL-Tutor

5.3 Constraint-based Student Modeling to Facilitate Case Based Design of Positive Feedback

One major output of this research was to consider how constraint-based modelling could be used to facilitate the scheduling of positive feedback. Two types of positive feedback moves were initially defined:

- When the student is expressing uncertainty but nevertheless does the right thing
- When the student is too paralyzed to do anything at all and requests help

Examples of the first type of move and some of the second are observed in tutoring protocols obtained from both the computer science domain (data structures) and the

letter sequence pattern extrapolation domain obtained from Stellan Ohlsson at the University of Illinois. Of interest to this research was the question of how ITSs could be made to identify and of course respond to these moves. The most basic approach of course would be to simply give positive feedback after every correct student move. This however would not be too efficient or effective as students would undoubtedly learn to ignore these messages due to the sher quantity. A step up from this approach but totally different in nature, is to have students rate their level of certainty about their answers on a rating scale, simply accepting their own judgement of their uncertainty. This however introduces some degree of subjectivity as one students estimation will undoubtedly differ from another. This could however be managed.

On another hand one could experiment with response times inferring that the longer a student takes to make the next step, the more unsure or hesitant they are about the correctness of that step. One criterion for giving positive feedback would therefore be, long pause followed by a correct step. The major problem with this approach is that the learning environment is not constrained, and a student taking a break to go to the kitchen to get a cup of tea might be construed as a long pause and thus uncertainty. This will result in inconsistencies in modelling the student.

While in this research we will make use of response times, we are more interested in how the constraint base could support identifying those instances or cases when positive feedback should be given, that is, identifying uncertainty or lack of knowledge through student actions. Suppose for example that for a step X there are N relevant constraints, all satisfied but the last state in which some constraint C was relevant, it was violated. It might be that step X was the first time that the student is obeying constraint C and it would be reasonable to infer that the student has recently learned something new about which he might still have some residual uncertainty. It might be at this point that the system wishes to provide positive feedback. The student model therefore appears to have the information required to make adequate assessment of a student's uncertainty and in so doing build criteria for giving positive feedback.

In the sections that follow we show how the student model in combination with other techniques, can and has been exploited to develop a criteria for giving positive feedback.

5.4 Scheduling Positive Feedback - Designing Cases for Positive Feedback

In the previous chapter we discussed when it was positive feedback should be given. In the section before, we explain in simple terms how constraint-based student modeling can be used to identify student learning state. We now consider how the implementation of cases has been facilitated in SQL-Tutor through the use and analysis of constraints. We will discuss in detail the nine cases that were implemented:

- Case 1 - Constraint satisfied for the first time
- Case 2 - Constraint satisfied for the first time based on hint message automatically raised by the system after 5 minutes
- Case 3 - Constraint satisfied for the first time based on hint message requested by student
- Case 4 - Constraint satisfied after the student has either forgotten a constraint or is uncertain
- Case 5 - Constraint satisfied after the student has either forgotten a constraint or is uncertain and the system automatically generates a hint message
- Case 6 - Constraint satisfied after the student has either forgotten a constraint or is uncertain and an hint message is requested
- Case 7 - Difficult problem completed on the first attempt
- Case 8 - Difficult constraint satisfied for the first time

- Case 9 - Completing a problem

Case 1 generates positive feedback for each constraint satisfied for the first time (i.e. all previous attempts on that constraint were incorrect). Case 2 provides positive feedback when the student was given a hint on how to solve the problem by SQL-Tutor after a period of inactivity, and has managed to apply the hint successfully (i.e. the student satisfied the constraint for the first time in its history). In that case, the student will be given positive feedback on the newly satisfied constraint. Case 3 is similar to the previous rule, but covers the situation when the student requested a hint, rather than being provided with one automatically. Case 4 provides positive feedback on a constraint that a student used correctly earlier in the session, but then kept violating. When the student uses that constraint correctly again, case 4 would generate reinforcing positive feedback on that constraint. Case 5 covers a situation when a student was given a hint, and has consequently satisfied a constraint which was previously violated. Case 6 covers a similar situation, but differs from case 5 in the fact that the student requested a hint. Case 7 generates positive feedback in the case of a difficult problem being solved correctly, while case 8 provides positive feedback after satisfying a difficult constraint. We consider these cases in even greater detail in the section that follows. Please note that cases are presented not in the order in which they are applied but under the generic category from which they were derived. Detailed examples will not be presented for cases 3 and 4 as detailed examples are present for cases 5 and 6. These sets of cases are similar in that they both have as triggering conditions, an hint message requested and hint message automatically generated. It is therefore not necessary to repeat examples.

5.4.1 Positive Feedback Under Uncertainty

In this research we consider uncertain student actions to be those actions taken by a student in which there exist some degree of doubt. When attempting a problem task, students can usually be rated on what can be referred to as a scale of doubt, each student having some level of uncertainty (ranging from none to absolute) associated with their

actions to solve the required problem or problem steps. If positive feedback can be given precisely when a student is showing signs of uncertainty, then we hypothesize that it will reduce that uncertainty and increase learning rate and potentially performance. Below we present the criteria we use to identify cases of uncertainty and give examples of its implementation.

We begin our analysis with case 1 which refers to the first time there is evidence that a constraint has been learnt. We consider a situation where a student is uncertain or simply does not know the correct format for the date within SQL. If the student gets the format wrong, the first time or a number of times after that, then we are certain of the student's knowledge of SQL date formats. It is either they are uncertain of the format, maybe toggling between two options or simply do not know and is guessing. In either case the student has submitted an incorrect answer and the constraint(s) ensuring correct date format is violated.

In this case, choose to give feedback the next time the constraint is relevant and the student satisfies it. Based on our hypothesis, if the student is uncertain, that is, toggling between a series of options one of which may be correct, that is choosing from several chunks of knowledge of problem strategies trying to find the correct one or simply guessing positive feedback will help reinforce those knowledge chunks which led to the student obtaining the correct answer.

We consider below a detailed example of case 1 as is implemented in SQL-Tutor:

Case 1

> *If X is a relevant constraint and in the last step, or series of steps X was violated (that is X has never been satisfied), then give positive feedback the next time X is relevant and satisfied.*

Case 1

SQL tutor problem 264

Question: List book titles and the quantity held in stock.

Student Solution (1)

The student attempts the problem the first time submitting the following solution as the completed solution:

```
Select title, quantity
From Book, Inventory
```

Summary

As a result constraint 13 is violated (see Code 3). From a review of the constraint we note that constraint 13 was relevant because the student specified no join conditions. The student has used several tables in this query and needs join conditions that specify how the data from the tables is to be combined. A join condition compares the values of two attributes, coming from two tables. In most cases, if there are N tables in the query, then there should be N-1 join conditions. If the primary key contains more than one attribute, all attributes of the primary key must appear in join conditions.

Case 1 cont'd

Student Solution (2)

The student based on feedback or otherwise makes amendments to his/her previous submission and resubmits, this time correcting the previously violated constraint:

Select title, quantity
From Book, Inventory
Where Book.Code = Inventory.Book

Give positive Feedback

We apply the rule above (1) and since constraint 13 is a relevant constraint which was violated in the last step, but is currently satisfied, we give positive feedback.

Feedback Message

Great work! You have specified all the necessary join conditions.

Case 4

Case 4 is a more general case where the student has previously encountered a constraint and satisfied it either for the first time or on several occasions. However it might be that the student has simply guessed the answer or has been applying faulty knowledge in such a way that they have been successful but only up to given point. An example of the later could be addition of fractions where one denominator is a multiple of the other. Choosing the larger denominator in this case would result in the calculation of

Code 3 SQl-Tutor Constraint 13

```
'(p 13
   "Check whether you have specified all the necessary join
   conditions."
    (and (> (length (find-names2 ss 'from-clause)) 1)
         (not (member "JOIN" (from-clause ss) :test 'equalp))
         (not (member "SELECT" (where ss) :test 'equalp))
         (or (member "JOIN" (from-clause is) :test 'equalp)
    (> (length (join-cond-is (slot-value is 'where) '())) 0))
         (equalp (length (find-names2 ss 'from-clause))
           (single-att-keys (find-names2 ss 'from-clause) 0)))
    (equalp (- (length (find-names2 ss 'from-clause)) 1)
    (length (join-cond-ss (slot-value ss 'where) '())))
    "WHERE")
```

the correct common denominator, however if it is not the case that one denominator is a multiple of the other, then this approach will not work. This is a case where the student knowledge or application thereof is incomplete and is adequate for special cases but not for all. A student may therefore proceed correctly without violating a given constraint but a situation will eventually arise whether after 2 or more subsequent steps where the constraint is violated. If the student gains or infers the knowledge required to satisfy the constraint, that is, satisfies the constraint the next time it is relevant, we propose that positive feedback be given.

> *If X is a relevant and satisfied constraint in one step but violated in the second or several subsequent steps, then give positive feedback the next time X is relevant and satisfied.*

Case 4

SQL tutor problem 268

Produce a list of book titles and authors. Order it by authors' last names.

Student Solution (1)

The student attempts the problem the first time submitting the following solution as the completed solution:

Select title, lname, fname
From Book, Author
Where author.authorid = writtenby.author and book.code = writtenby.book
Order by lname

Summary

As a result constraints 10, 13 and 7 (see Code 4) are violated. From our previous example (case 1) above we note that constraint 7 was a relevant but satisfied, that is, all the names used as attributes names in the where clause where attribute names from the tables named in the from clause.

Student Solution (2)

The student based on feedback or otherwise makes amendments to his/her previous submission and resubmits, this time correcting the previously violated constraint:

Select title, lname, fname
From Book, Author, Writtenby

Case 4 cont'd

Where author.authorid = writtenby.author and book.code = writtenby.book
Order by lname

Give Positive Feedback

We apply the rule above (4) and since constraint 7 is a relevant constraint which was satisfied in a previous problem, but was violated in the last step and is currently satisfied, we give positive feedback.

Feedback Message

Thats correct! All the names used as attributes names in the WHERE clause must be attribute names from the tables named in the FROM clause.

5.4.2 Positive Feedback Under Paralysis

We consider in case 6 and also in case 5 below, the situation where despite error-specific feedback provided by SQL-Tutor, the student does not obtain the correct solution. Even when negative feedback is given, the information provided does not seem to assist the student in any way and the student is no better off than previous. Continuous attempts at obtaining correct solutions result in violated constraints. It could possibly be that the student might be oblivious as to the next step or lacks the procedural and/or declarative knowledge required to arrive at a solution. It could also be that the student lacks some

Code 4 SQl-Tutor Constraint 13, 10 and 7

```
'(p 10
   "You do not have all the tables that are needed
   in the FROM clause!"
   (null (member "SELECT" (where is) :test 'equalp))
   (null (list-difference (all-tables-used is)
   (all-tables-used ss)))
   "FROM")

'(p 13
   "Check whether you have specified all the necessary join
   conditions."
    (and (> (length (find-names2 ss 'from-clause)) 1)
         (not (member "JOIN" (from-clause ss) :test 'equalp))
         (not (member "SELECT" (where ss) :test 'equalp))
         (or (member "JOIN" (from-clause is) :test 'equalp)
    (> (length (join-cond-is (slot-value is 'where) '())) 0))
         (equalp (length (find-names2 ss 'from-clause))
         (single-att-keys (find-names2 ss 'from-clause) 0)))
    (equalp (- (length (find-names2 ss 'from-clause)) 1)
    (length (join-cond-ss (slot-value ss 'where) '())))
    "WHERE")

'(p 7
   "All the names used as attributes names in the WHERE clause
   must be attribute names from the tables named in the FROM
   clause."
    (and (not (null (where ss)))
         (not (member "SELECT" (where ss) :test 'equalp))
         (bind-all ?n (find-names ss 'where) bindings))
    (attribute-in-from ss ?n)
    "WHERE")
```

insight knowledge required to successfully solve the problem. In either situation the student is too paralyzed to do anything and can only move forward through the assistance of the tutor in the form of some more direct hint on what the solution should be.

If indirect feedback is given, that is the student is given a hint to the answer but not the answer itself, and the student successfully uses this added knowledge to correctly infer the answer, it shows that the student had at least some knowledge of the problem requirements which they successfully applied and positive feedback should be given. Based on our hypothesis positive feedback should increase student confidence. It will also decrease uncertainty over the student's existing knowledge, decrease uncertainty over the knowledge gained from receiving the hint statement which was once lacking and strengthen the connection between these two pieces of knowledge chunks.

Case 6

> *If the student is too paralyzed to take action, that is, the student is at a point in the task where he or she is totally and absolutely unsure of what is required and the student requests a hint; and if on the next step subsequent to the hint, the student corrects a previously violated constraint based on the information provided in the hint, that is, the constraint is linked to the hint message given, then provide positive feedback.*

Case 6

SQL tutor problem 267

List the book titles and publisher's names for books that have 1 or less copies in stock.

Student Solution (1)

The student attempts the problem the first time submitting the following solution as the completed solution:

Select title, publisher
From Book, Inventory
Where Quantity $<=$ 1

Summary

As a result constraint 10, 13 and 650 are violated (see Code 6). Constraints 10 and 13 have already been presented and discussed. Constraint 650 which fires when all required attributes have not been included in the select clause has also been violated. At this point the student is being given simple and error flag feedback and might feel paralyzed not knowing what the error is or where to correct it. The student then decides to receive a hint from the system and so clicks the hint button to generate a hint message as follows:

Hint Message

You do not have all the tables that are needed in the FROM clause!

Student Solution (2)

The student based on the hint message feedback given, makes amendments to his/her previous submission and resubmits, this time

Case 6 cont'd

correcting the previously violated constraint:

Select title, publisher

From Book, Inventory, Publisher

Where quantity $<= 1$

Give positive Feedback

We apply the rule above (6) and since the student has requested a hint and on the subsequent step corrected a previously violated constraint using information obtained in the hint (i.e. the hint message received is linked to constraint 10 previously violated and now corrected), then we provide positive feedback.

Feedback Message

Well done! You now have all the tables needed in the FROM clause.

Code 5 SQl-Tutor Constraint 650

```
'(p 650
   "You do not have all the required attributes in the
   SELECT clause!"
    (bind-all ?n (names (select-clause is)) bindings)
    (my-member ?n (names (select-clause ss)) is)
    "SELECT")
```

Case 5

This case is similar to case 6 with the exception that the hint message or assistance provided is decided upon and generated by the system rather than the student. It targets the situation where the student needs help but has not requested help or had any interaction with the system for some fixed period of time. Admittedly we do not expect to see too many occurrences of this case as most students will seek help when needed either through the system or externally.

> *If the student is too paralyzed to take action and 5 minutes elapses without an attempt to the solution, provide a hint to the student and if student on the next subsequent step corrects a previously violated constraint based on the information provided in the hint message, provide positive feedback.*

Case 5

SQL tutor problem 263

Give the titles of books written by author whose id is 20.

Student Solution (1)

The student attempts the problem the first time submitting the following solution as the completed solution:

Select title
From Book
Where authorid = 20

Constraints Violated

As a result constraint 10, 7, 166 and 427 (see below) are violated.

Case 5 cont'd

Student Solution (2)

The student's second submission is:

Select title

From Book, Writtenby

Where authorid = 20

Constraints Violated

As a result constraint 476, 7, 166 and 427 (see below) are violated:

Student Solution (3)

After making changes, the student makes the following submission:

Select title

From Book, Writtenby

Where book.code = writtenby.book and authorid = 20

Constraints Violated

As a result constraint 7, 166 and 427 (see) are violated.

Summary

After several attempts the student has not yet used a hint and has not yet solved the problem. The student came close to the answer after the third attempt but still had three constraints being violated. These included constraint 7 previously discussed, constraint 166 which states that you need a search condition in WHERE, specified with a numerical constant has also been violated and finally constraint 427 which requires the search condition specified in the WHERE clause to compare the value of the attribute to an integer. On the fourth submission and

Case 5 cont'd

after 5 minutes of making no attempt at the problem, the system has submitted a hint message to assist the student and they have satisfied the constraint. *Hint Message*

Check that the attribute you have used to compare to the integer value in the WHERE clause is correct!

Student Solution (4)

The student after five minutes and receiving the hint message makes the following correct submission:

Select title

From Book, Writtenby

Where book.code = writtenby.book and author = 20

Give positive Feedback

We apply the rule above (5) and provide positive feedback because the student has successfully used the system generated hint to satisfy the constraint previously violated.

Feedback Message

Well done! All the names used as attributes names in the WHERE clause must be attribute names from the tables named in the FROM clause. "AUTHOR" is a valid attribute name in the FROM clause.

Code 6 SQl-Tutor Constraint 166, 427 and 476

```
'(p 166
   "Do you need a search condition in WHERE, specified with a
   numerical constant?"
   (and (not (null (where ss)))
        (bind-all ?n (integers-only (where ss)) bindings)
        (match '(?*d1 (?is ?a valid-attribute)
                      (?or "=" "<" ">" "<=" ">=" "<>" "!=") ?n ?*d2)
               (where ss) bindings)
        (null (qualified-name ?a)))
   (member ?a (where is) :test 'equalp)
   "WHERE")

'(p 427
  "Check the search condition you specified in the WHERE clause
  to compare the value of the attribute to an integer!"
 (and (not (null (where ss)))
      (not (null (where is)))
      (bind-all ?n (names (where is)) bindings)
 (attribute-in-db (find-schema (current-database *student*)) ?n is)
 (match '(?*d3 ?n "=" (?is ?c1 my-integerp)?*d4)(where is) bindings)
 (match '(?*d1 ?n1 (?is ?op rel-p) ?c1 ?*d2) (where ss) bindings))
 (and (same-attributes ?n1 ?n)
      (equalp ?op "="))
    "WHERE")

'(p 476
   "Check whether you have specified all the necessary join
   conditions."
    (and (> (length (find-names2 ss 'from-clause)) 1)
         (not (member "JOIN" (from-clause ss) :test 'equalp))
         (not (member "SELECT" (where ss) :test 'equalp))
         (or (member "JOIN" (from-clause is) :test 'equalp)
         (> (length (join-cond-is (slot-value is 'where) '())) 0))
         (not (equalp (length (find-names2 ss 'from-clause))
         (single-att-keys (find-names2 ss 'from-clause) 0)))
    (zerop (additional-joins (find-names2 ss 'from-clause) 0)))
    (equalp (- (length (find-names2 ss 'from-clause)) 1)
       (length (join-cond-ss (slot-value ss 'where) '())))
    "WHERE")
```

5.4.3 Positive Feedback Generally

Below we describe the more general cases not directly related to student uncertainty when we feel positive feedback should also be given. These cases are self-explanatory and presented below:

Case 7

If the student in three or fewer attempts gets correct a problem identified as difficult, that is, a problem of high complexity, then provide positive feedback.

Case 7

SQL tutor problem 194

We consider the following problem rated at a complexity level of 7:
Find the ids of artists who recorded every song on the CD titled 'The Distance to Here'.

Student Solution

The student attempts the problem finally submitting the following solution as the completed solution on the third attempt:

```
SELECT artist
FROM performs
WHERE not exists (select id
from recording
where not exists
```

Case 7 cont'd

(select ∗
from contains,cd
where contains.cd=catno and id=rec
and title='The Distance to Here'))

Summary

The student violates a number of constraints which we do not consider in detail for purposes of this example. The student makes no more than three attempts and correctly solves the problem on the third attempt.

Give positive Feedback

We apply the rule above (7) and since the student has completed correctly a problem of high complexity (7), this represents a significant achievement and we therefore give positive feedback.

Feedback Message

Fantastic, you've solved a highly difficult problem in very few attempts, choose another problem.

Case 8

If the student satisfies for the first time a constraint identified as difficult, that is, a constraint testing a concept of high complexity, then provide positive feedback.

Case 9

When the student gets the entire problem correct then that represents a significant achievement, therefore give positive feedback for problem the student gets correct.

Case 9

SQL tutor problem 270

We consider the following problem rated at a complexity level of 5:
How many books are paperbacks?

Student Solution (1)

The student attempts the problem the first time submitting the following solution as the completed solution:

Select count (code)
From Book
Where paperback = 't'

Summary

No constraints have been violated. The student has completed correctly a level 5 complexity problem on the first submission without the use of any feedback. Also the student has completed the select clause without error. The solution for this clause involves the use of a count function and this commonly causes errors among student.

Case 9 cont'd

Give positive Feedback

We apply the rule above (9) and since the student has successfully completed a problem we provide positive feedback. Alternatively if the student had not completed the problem correctly but had correctly submitted for the select clause on the first attempt then we provide positive feedback under rule (8) satisfying a difficult constraint on the first attempt.

Feedback Message

Your solution is correct - well done!

5.5 Selection and Specification of the Content of Positive Feedback

The content of positive feedback designed for this study contained two components, the verification and elaborative components. The verification information differed depending on the feedback case being generated by the system based on the student model. The following is an explanation of the verification information content presented and the case which generated it.

- "Well done." - presented if a student was receiving feedback in response to a hint message received and correctly processed.

- "Fantastic work." - presented in the case where a difficult constraint is satisfied for the first time.
- "Great work." - presented when a constraint is satisfied for the first time.
- "That is correct." - presented when the student corrects a previously violated constraint.

The second step was to generate elaborate information content. In SQL-Tutor, constraint-based tutor, feedback messages are generated logically from constraints which effectively partition the domain. Consider the fraction domain and the addition of fractions. The property "a numerator equal to the sum of the numerators of two unequal fractions" defines an equivalence class of problem states, namely the set of all states that contain such a numerator and divides the universe of possible solution paths into two disjoint classes. Likewise the property "every SQL select statement must contain a FROM clause" defines the set of all problem/solution states that specify the mandatory FROM clause. These properties can be translated into constraints and visa versa. It is in fact the translation of these fundamental ideas and basic principles which constitute modeling of the domain.

The elaboration information content for positive feedback was therefore developed in line with negative feedback content translating logical constraints used to model the domain into descriptive narratives which reveal to the student the domain concepts, principles and properties together with response specific and also topic specific information to reveal the underlying concept modeled by the given constraint.

Consider the following example where the student receives a positive feedback message in response to violating a constraint on the FROM clause of the SELECT statement. The student has failed to specify a table name or any other information for this clause of the statement. This violates a basic principle of the SQL query language which specifies that the FROM clause of an SQL SELECT query is mandatory, that is, the tables from which information is to be retrieved must be specified in the FROM clause. The

elaboration information provided would therefore read for example, “The FROM clause is a mandatory one.” Combining these two components, the full feedback message produced and rendered to the student would read, “That is correct. The FROM clause is a mandatory one.”.

Each message was developed tailored to each individual constraint. To improve on the feedback message constraints were instantiated vis-a-vis the problem situation so as to make the feedback message more life-like and personalized, providing the facility to respond to each student specifically. This involved retrieving the variable bindings from the constraints relevant and satisfied and translating them into English.

Consider constraint number 454 related to specifying attributes in string comparison operations. The positive feedback message for this constraint would read as follows: “You have specified the right attribute *bound variable* to compare the string constant to in WHERE.”. When instantiated the *bound variable* in the message would be replaced by the actual attribute used by the student.

5.6 Selection and Specification of the Form or Modus of Positive Feedback Presentation

No changes were made to SQL-Tutor in the method used to present feedback messages. As mentioned before, the same methods were applied to both positive and negative feedback messages which were presented in the same manner. All messages were presented in black text on the right side of the middle frame against a grey background. The only difference in the feedback messages would have been the content.

CHAPTER 6

Evaluation

6.1 Description

An evaluation study was conducted at the University of Canterbury from May 9, 2007 to June 6, 2007 during the second term of 2007. The evaluation study was conducted using SQL-Tutor, the intelligent tutoring system for tutoring SQL. Students at the University of Canterbury are taught the structured query language (SQL) in a 200-level database course (COSC 226) offered within the Computer Science and Software Engineering undergraduate degree structure. As part of the course, students were required to attend weekly lectures where they were taught the relevant theory. Amongst other tools, students could use SQL-Tutor to practice their skills in either the weekly lab sessions or at their leisure; such use of SQL-Tutor was strictly on a voluntary basis.

6.2 Participants

The participants for the evaluation study were students from the COSC 226 course, an undergraduate database course at the University of Canterbury. The tutoring system was made available for use by students of the course who could decide to use the system

voluntarily over a given period of time. Seventy nine (79) students enrolled in the course. Of these students, 55 logged into SQL-Tutor at least once and 51 completed the pretest. Participants were divided into two groups: the control and the experimental.

6.3 Method

Participants were allowed to log into SQL-Tutor for the first time during scheduled laboratory sessions starting from May 9, 2007. Participants could also log into the system outside of these hours both on site at the university or off site, for example from their own house. COSC226 Students were allocated user accounts in advance. The login module within SQL-Tutor assigned students to either the control group or the experimental group; each student remaining in their assigned group during the entire evaluation period.

At first login, users were presented with one of two online pretests. On completion of the pretest, users continued onto their first problem. On a pre-specified date approaching the end of the evaluation period (June 6, 2007), students were presented with one of two online post-tests. These were automatically assigned to anyone logging unto the system on or after the end of the evaluation period. The pre and post-tests were tests created and used in previous studies of SQL-Tutor (See Appendix B). Each test contained four multi-choice questions of comparable complexity. The marks for both tests were recorded and stored in student log files.

As students used the system, their constraint and problem histories were recorded in individual student models. Student actions and any decisions made by the ITS were also recorded in individual student logs. For the experimental group, this also included the history of positive feedback messages received and the positive feedback cases triggered during the students tutoring session.

When selecting a new problem, users were first presented with a choice of SQL clauses and in each case the system choice was selected by default. If users selected a choice other than the system choice, they were presented with a confirmation dialogue,

alerting them to their deviation from the system choice. Users were next presented with a set of problems appropriate to their selection and their knowledge level. In this set, the system problem choice was also highlighted again with the possibility of an override if desired by the user (See Appendix C). After selecting the problem, the user was presented with the problem text and solution area where feedback would be displayed. In both versions (control and experimental), the user could submit the solution at any stage. Six levels of feedback, ranging from simple knowledge-of-response feedback to knowledge-of-correct-response (full solution), were available to the user.

6.4 Results and Discussion

6.4.1 Pre and Post-test

A total of 55 students from the COSC 226 database course took part in the evaluation study over the period May 9, 2007 to June 6 2007 logging into SQL-Tutor which randomly assigned 30 to the control group and 25 to the experimental group. Both groups were required to sit pre and post-tests to assess their initial understanding of the domain principles, and subsequently to measure any improvements or learning gains. Both the pre and post-tests contained four problems, each worth one mark; thus a student could gain a maximum of four marks for either of the tests. Of the 55 students, 4 logged into the system without having any use at all while 51 attempted the pretest. Of that 51, 37 attempted both the post test and the pretest. It was found that everyone who attempted the post-test also attempted the pretest.

In the analysis, students spending less than 10 minutes and with 6 or fewer attempts were excluded as it was assumed that this was insufficient time to produce learning effects in the system. This was confirmed as these students also fitted the criteria of having learnt zero constraints. After eliminating these students we were left with 42 who attempted the pretest and 34 who attempted both the pre and post-test. Taking only those students attempting both the pretest and post-test, we assessed the mean and

standard deviations for the two groups. Table 6.1 below shows the results of the pre and post-tests for the 34 students.

	Control	**Experimental**
Number of participants	19	15
Pretest	1.8 (0.9)	2.2 (1.1)
Post-test	1.5 (1.2)	1.7 (1.3)

Table 6.1: Analysis of Pretest and Post-test Results (1)

To ensure there was no significant difference between the groups at pretest, an ANOVA analysis was done on pre and post-test scores. There was no significant difference between the two groups on the pretest performance($p = 0.2$) and no significant difference on the post-test results (see Table 6.1)($p = 0.32$). Mean post-test scores were however much lower than pretest scores and investigations revealed limitations in the recording of pre and post-test scores. These were as a result of two factors: firstly, the post-test was not mandatory until the end of the study (June 6, 2007), and secondly post-test scores were not recorded accurately. It was mandatory that students see and submit the pretest before being allowed to proceed through the tutor, however due to the inherent nature of the study, students were allowed to use the system without having to see or attempt the post-test until the specified post-test date. The second factor related to the recording of scores and was due to a technical limitation inherent in both versions of SQL-Tutor. The system did not record clearly a student giving no answer versus a student giving an incorrect answer. In both cases the answer to the question was effectively recorded as an incorrect answer, that is, a zero score was given. This therefore meant that students not even attempting the post-test were given a zero score and could not be differentiated from those who actually attempted and received a zero score. We believe that this adequately explains post-test results which when taken together and also separately for each group, were very low compared to pretest scores.

A greater number of students attempted the pretest compared to the post-test. While 24 of the 25 students in the experimental group attempted the pretest, only 16 attempted the post-test. 27 of the 30 students in the control group attempted the pretest while only 21 attempted the post-test. In the control group there was one zero score on pretest versus five on the post-test. In the experimental group two students had zero scores on the pretest while five scored zero on the post-test (see Table 6.2). These factors lead to reduced confidence placed on post-test scores and account for what we believe are very low post-test scores compared to pretest. Many students either did not attempt or most probably did not pay strong attention to the post-test. Only 50% of students who attempted the post-test scored 1 or above, while 92.7% of students who attempted the pretest scored at least 1. It is for this reason that throughout the rest of the analysis we place reliance on pretest scores but none on post-test.

	Control	**Experimental**
Number of participants	30	25
Attempting Pretest	27	24
Attempting Post-test	21	16
Zero Score on Pretest	1	2
Zero Score on Post-test	5	5

Table 6.2: Analysis of Pretest and Post-test Results (2)

With this in mind analysis was continued involving the 42 students who spent greater than 10 minutes in the system and had more than 6 problem attempts. This involved initially some descriptive statistics which revealed a significant outlier in the experimental group and this student was removed, leaving 41 students whose data we believed was a good representation of both groups. T-test analysis on pretest scores was repeated for the remaining 41 students and it showed no significant difference in means ($p = 0.27$). This meant that prior knowledge as measured by the pretest was not a significant factors across the groups. The summarized statistics for these students are presented in Table 6.3.

	Control	Experimental	p
Number of Participants	23	18	ns
Pretest mean (sd)	1.7 (0.8)	2.1 (1.3)	ns
Constraints Learned	10.0 (6.1)	9.3 (6.8)	ns
Time (min)	193.8 (198.7)	92.3 (44.7)	0.012
Problems Solved	25 (24)	22 (15)	ns
Total Problem Attempts	119 (99)	98 (66)	ns
Problems Attempted	28 (25)	26 (15)	ns
Lab-test mean (%)	57.0 (26.5)	59.3 (24.3)	ns
Average Time per Solved Problem	9.8 (7.9)	5.8 (4.8)	0.024
Average Time per Attempted Problem	7.5 (4.5)	4.1 (2.0)	0.002
Attempts per Problem	4.5 (1.6)	3.7 (0.9)	ns
Attempts per Solved Problem	5.5 (2.9)	5.2 (3.4)	ns

Table 6.3: Summary Analysis of Students Interaction with SQL-Tutor

Overall and at first glance, the experimental group appears to have performed better than the control group. The experimental group appears to have solved almost the same number of problems in significantly less time, made fewer attempts while attempting on average almost the same number of problems as the control group. They also learned almost the same number of constraints. Additionally the experimental group's mean performance on the lab-test was slightly higher. The lab-test compared to the post-test and pretest is much longer and used as a practical assessment for the course and tests not only students ability to generate SQL queries but their ability to define, update, populate and manage databases. It is an objective measure of the students understanding of the practical aspects of the course. To substantiate and support these initial findings a number of standard statistical tests were applied and models created.

We analyzed the difference in means for the two groups for each of the measures reported in Table 6.3 using the t-test. The only significant differences noted were for the time spent with SQL-Tutor ($p = 0.012$), average time per problem solved ($p = 0.024$) and the average time per attempted problem ($p = 0.002$). What was a more significant or noticeable result of the study, was the time spent working with the system versus the number of problems solved. The mean time spent using SQL-Tutor for the experimen-

tal group was half that of the control group. While students in the experimental group spent on average 101.5 minutes less than students in the control group, they were able to solve almost the same number of problems; an average of 22 solved problems in the experimental group compared to an average of 25 by students in the control group. We also compared students' performance on a lab test, which was an assessment item for the course, performed after the SQL-Tutor study. The experimental group's mean performance on the lab-test was slightly higher, although the difference is not significant.

To further investigate the effects of the two groups on time, an ANCOVA analysis was performed with time spent as the dependent variable and mode (group membership), pretest, number of negative feedback messages seen, number of solved problems and total attempts as explanatory variables. All factors together accounted for 86% ($R^2 = 0.862$) of the variability in total time spent and an interaction was noticed between mode and total time spent in the system ($p = 0.005$); students in the experimental group receiving positive and negative feedback spent less time in the system, than students in the control group who received only negative feedback and as observed previously this difference was significant ($p = 0.005$). Also significant was the total number of attempts ($p < 0.0001$). It is worth noting that prior knowledge as measured by pretest was not a significant factor ($p = 0.392$).

A time versus problems solved regression analysis was performed to determine the rates at which students were solving problems in the different groups. The equation for the control group was $y = 0.1127x + 3.5526$ ($p < 0.0001$) while the equation for the experimental group was $y = 0.2955x - 4.8427$ ($p < 0.0001$), the experimental group having a steeper sloped curve compared to the slope of the control group (see Figure 6.1). Extrapolating using these equations we note that initially students in the control group appear to solve more problems than those in the experimental, however after approximately 50 mins of tutoring time, students in the experimental group quickly overtake those in the control group. When we consider the results after 190 minutes of tutorial time, students in the experimental group (see Figure 6.2) solve more than twice as many problems as students in the control group (51 versus 25 after 190 minutes)(see

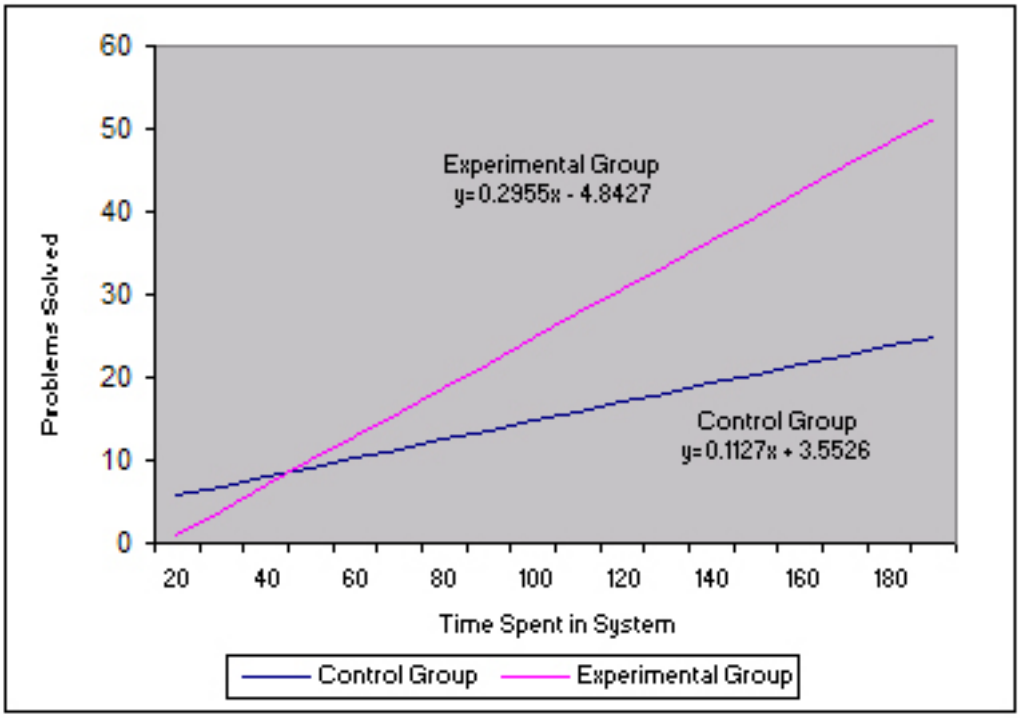

Figure 6.1: Comparing Experimental and Control Groups Regression Analysis: Solved Problems Versus Time Spent in System

Figure 6.3). Considering the interaction between group membership and time spent in the system and the difference in the mean time for the two groups, we see that students in the experimental group solved on average the same number of problems in less than half the time as those in the control group.

To further consider the impact of all factors on learning, we performed two multiple regression analyses. Of particular interest was the students prior knowledge as reflected in the pre-test score, the total time spent receiving tutoring and the number of feedback messages seen. The last factor consisted of the number of negative feedback messages seen by control group, while for the experimental group, we used the two factors: the number of negative and the number of positive messages. The number of learned constraints and lab-test scores were both used as a measure of learning and we discuss them separately because they reveal different and interesting results about the students interaction with the system. The method used for adding the variables was EN-

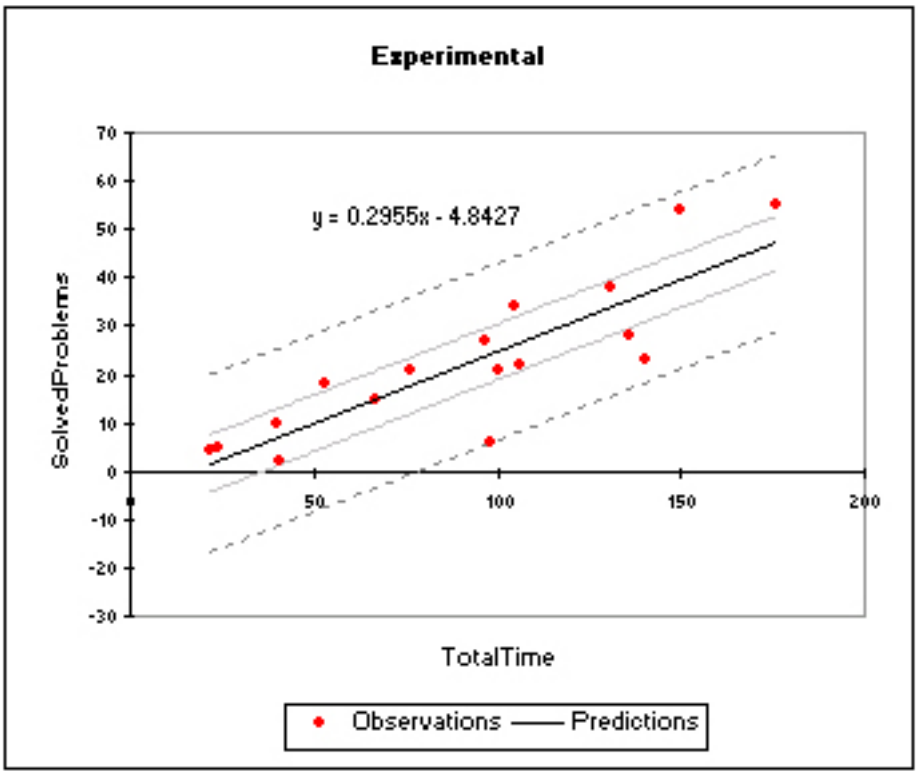

Figure 6.2: Scatter Diagram Showing Solved Problems Versus Time Spent in System for Experimental Group

TER and variables were added in the same order as given in the results. These results are shown in Tables 6.4 and 6.5. Note that the beta values shown are for the final regression model. We first consider the results of learning as measured by the number of learned constraints (Table 6.4) (Table note: ns - not significant). The number of learned constraints is an internal measure by the system based on a very simple heuristic. Each constraint is considered individually initially taking the first five attempts involving the constraint. If the student satisfies the constraint less than seventy percent (70%) of the number of times it is relevant, then the student is recorded as knowing the constraint otherwise the constraint is flagged as a constraint to be learnt. At the end of the session the last five attempts are taken and the same calculation performed. If the student previously did not know the constraint but has now satisfied the constraint more than seventy percent (70%) of the number of times its relevant then the constraint has been learnt.

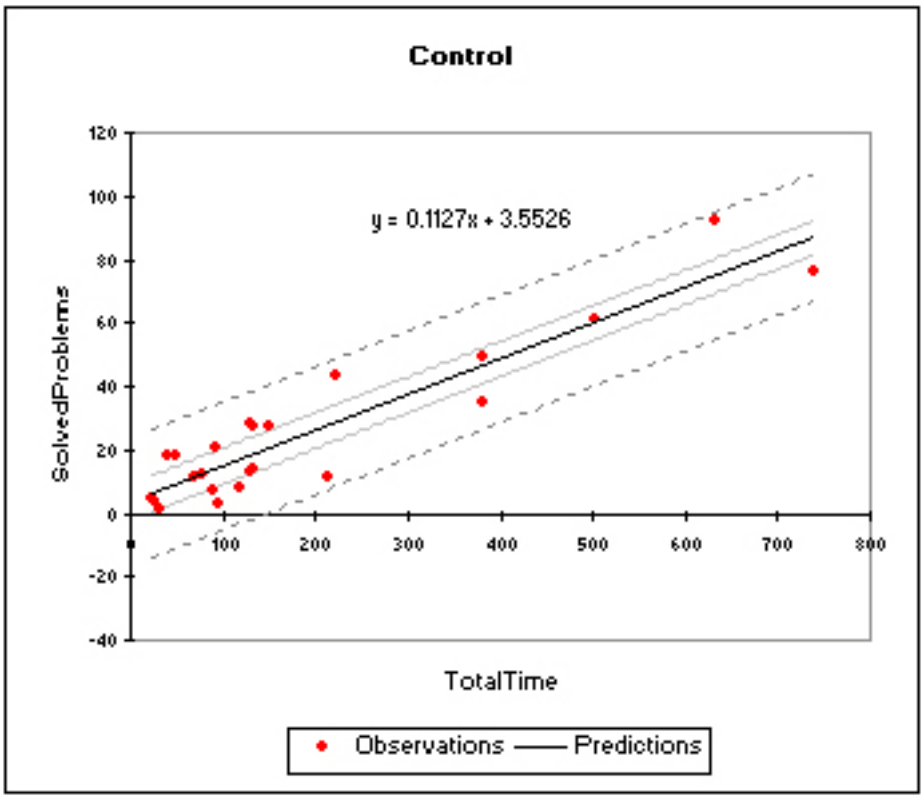

Figure 6.3: Scatter Diagram Showing Solved Problems Versus Time Spent in System for Control Group

The measure therefore only applies to newly acquired constraints or concepts, that is, those learned as a result of using the system.

For the experimental group, the model we obtained including all four predictors accounted for 77% of the variance, with time contributing most. The number of positive feedback messages is the only (marginally) significant predictor, and accounts for 6% of the variance. For the control group, the model with three predictors explains 71% of the variance, and the number of negative feedback messages is the only significant predictor. These results suggest that the students prior knowledge did not affect the average number of constraints learned and did not explain a significant portion of the variance in the learning for any of the groups. We see that the strongest predictor of learning in the case of the experimental group is the number of positive feedback mes-

sages seen, while in the case of the control group, the strongest predictor turns out to be the number of negative feedback messages seen. It is not surprising that negative feedback is the strongest predictor of learning, for this is a rather common finding in educational research; however it is interesting that positive feedback turns out to be the strongest predictor for the experimental group. It shows that student's learning is influenced considerably by positive feedback messages received and that the more positive feedback messages a student receives, the more they are likely to learn ($\beta = 0.625$). This evidence supports directly the hypothesis of this research; that positive feedback works by reducing uncertainty in student knowledge, decreasing the number of future errors by strengthening weak knowledge and in some cases creating new knowledge. The high correlation between learnt constraints and time also supports this, r=0.74 for the experimental and r=0.68 for the control group.

		R^2change	β
Control	**Pretest**	0.015 (ns)	0.083 (ns)
R^2=0.714	**Time**	0.458 $(p < 0.001)$	-0.285 (ns)
	Negative feedback	0.241 $(p = 0.001)$	1.080 $(p = 0.001)$
Experimental	**Pretest**	0.000 (ns)	-0.036 (ns)
R^2=0.766	**Time**	0.560 $(p = 0.001)$	0.075 (ns)
	Negative feedback	0.143 $(p = 0.021)$	0.203 (ns)
	Positive feedback	0.063 $(p = 0.083)$	0.625 $(p = 0.083)$

Table 6.4: Results from Multiple Regression, by Learned Constraint

We also performed a multiple regression of the lab-test scores using the same factors as a basis for comparison (Table 6.5). As previously mentioned, the lab-test is an independent measure of the student's knowledge of SQL. To conduct that analysis we first extracted from the exam submissions only those questions and marks pertaining to the SQL-Tutor syllabus. For both groups we then compared these scores to the number of learned constraints and in both cases the correlation between labtest scores and learned constraints was very low, 0.07 for the experimental and 0.08 for the control group. We therefore expected the multiple regression to support these initial findings.

		R^2change	β
Control	**Pretest**	0.134 $(p = 0.094)$	0.375 $(p = 0.096)$
R^2=0.185	**Time**	0.002 (ns)	-0.402 (ns)
	Negative feedback	0.050 (ns)	0.502 (ns)
Experimental	**Pretest**	0.569 $(p < 0.001)$	0.546 $(p = 0.096)$
R^2=0.763	**Time**	0.052 (ns)	0.394 (ns)
	Negative feedback	0.024 (ns)	-1.402 (ns)
	Positive feedback	0.118 $(p = 0.031)$	0.885 $(p = 0.096)$

Table 6.5: Results from Multiple Regression, by Lab-test

The models (containing all predictors) obtained account for 19% and 76% of the variance for the control and experimental group respectively. Unlike our previous findings with learned constraints, we found that student prior knowledge marginally significantly predicts lab-test scores for both groups. In the case of the control group, prior knowledge accounts for 13% of the variance. In the experimental group prior knowledge accounts for 57% of the variance but interestingly positive feedback is also marginally significant, accounting for 12% of the variance. Given the very low correlation between learned constraints and labtest scores, and the findings of the learned constraint multiple regression analysis, these findings are not at all surprising. They reveal that students sitting the lab-test are influenced strongly by other sources of knowledge outside of that taught by SQL-Tutor. Despite this however, positive feedback still appeared to be a significant factor again supporting our hypothesis that positive feedback increases learning.

6.4.2 Positive Feedback

We considered more closely the experimental group who received positive feedback. Recall that by positive feedback we mean feedback given in response to correct student actions versus negative feedback which was given in response to incorrect actions. Positive feedback was implemented through several cases outlined as follows:

- Case 1 - Constraint satisfied for the first time

- Case 2 - Constraint satisfied for the first time based on hint message automatically raised by the system after 5 minutes
- Case 3 - Constraint satisfied for the first time based on hint message requested by student
- Case 4 - Constraint satisfied after the student has either forgotten a constraint or is uncertain
- Case 5 - Constraint satisfied after the student has either forgotten a constraint or is uncertain and the system automatically generates a hint message
- Case 6 - Constraint satisfied after the student has either forgotten a constraint or is uncertain and an hint message is requested
- Case 7 - Difficult problem completed on the first attempt
- Case 8 - Difficult constraint satisfied for the first time
- Case 9 - Completing a problem

Table 6.6 below shows the average number of positive feedback messages received by students in the experimental group and the distribution of these messages based on each case triggered. These figures show that students received most of their feedback from satisfying constraints the first time. This was followed not far behind by positive feedback given based on uncertainty and then positive feedback based on difficult constraints satisfied for the first time.

Analysis showed the mean number of negative feedback messages (182) received by the experimental group to be lower than that received by the control group (222) (see Table 6.7). One-way ANOVA on negative feedback messages revealed no significant difference in the mean number of negative feedback messages seen ($p = 0.9$) across the two groups. Expectedly, regression analysis revealed a linear relationship between total time spent, number of positive feedback messages and number of negative feedback

Case Description	Average number of messages
Case 1	8.5
Case 2	0.0
Case 3	0.8
Case 4	5.9
Case 5	0.0
Case 6	1.1
Case 7	1.0
Case 8	3.4

Table 6.6: Summary of Positive Feedback Messages Categorized by Case

messages seen ($R^2 = 0.698$) by students in the experimental group. Also in line with our hypothesis and experimental design is the mean number of negative feedback messages seen being considerably higher than the mean number of positive feedback messages.

	Number of Participants	Negative Feedback Messages	Positive Feedback Messages
Control	23	222 (169)	-
Experimental	18	182 (127)	22 (12)

Table 6.7: Positive and Negative Feedback in SQL-Tutor

This is expected because positive feedback is primarily in response to correct responses and is based on the system's measure of the student's uncertainty, a measure expected to decrease with time and the number of positive feedback messages received. A model ($p < 0.0001, R^2 = 0.6$) was then created using ANCOVA analysis with learnt constraint as a dependent variable and pretest, mode and negative feedback as explanatory. Results of the model revealed no significant interaction between learnt constraints and mode or pretest but as expected there was a very significant interaction between negative feedback and learned constraints ($p < 0.0001$).

6.4.3 Learning Curves

One method of analyzing whether the objects measured are related to the concepts learned is to plot learning curves. For this, the number of times the object (in our case, the constraint) is relevant was plotted against the proportion of times it was used incorrectly. If the object measured is being learned, a power curve should result (cited in [Mitrovic and Martin 2005]). Learning curves indicate whether the concepts taught in the domain have been learned. For more information on learning curves, see [Mitrovic and Martin 2004].

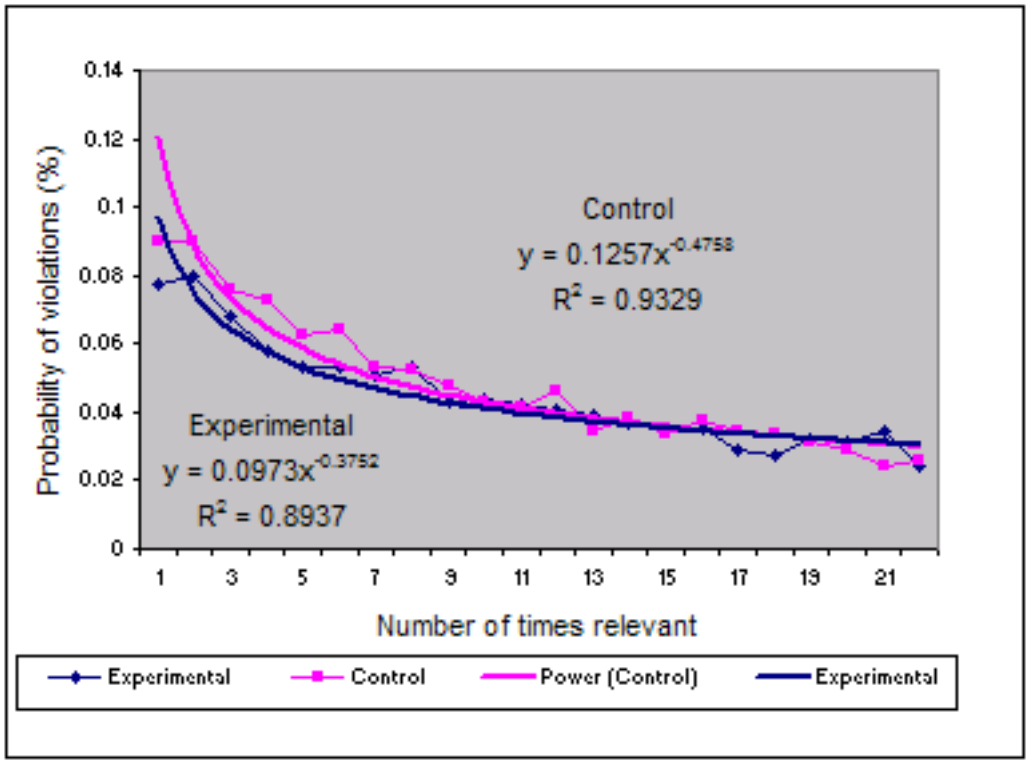

Figure 6.4: The Probability of Constraint Violations for all Constraints

Figure 6.4 shows the learning curves for both the control group and the experimental group for all constraints. The power curve equation for the control group was $y = 0.1257x^{-0.4758}$ and for the experimental group it was $y = 0.0973x^{-0.3752}$. It must be pointed out that these curves show the error rate averaged over all constraints and

all students, for each occasion when constraints were used. The actual data are well approximated by the power curves. Based on the slope of the learning curve, there appears to be no significant difference between the learning rates of the two groups, which is consistent with previous findings about the number of constraints learned. Students in the control group simply needed more time to learn the same amount of knowledge.

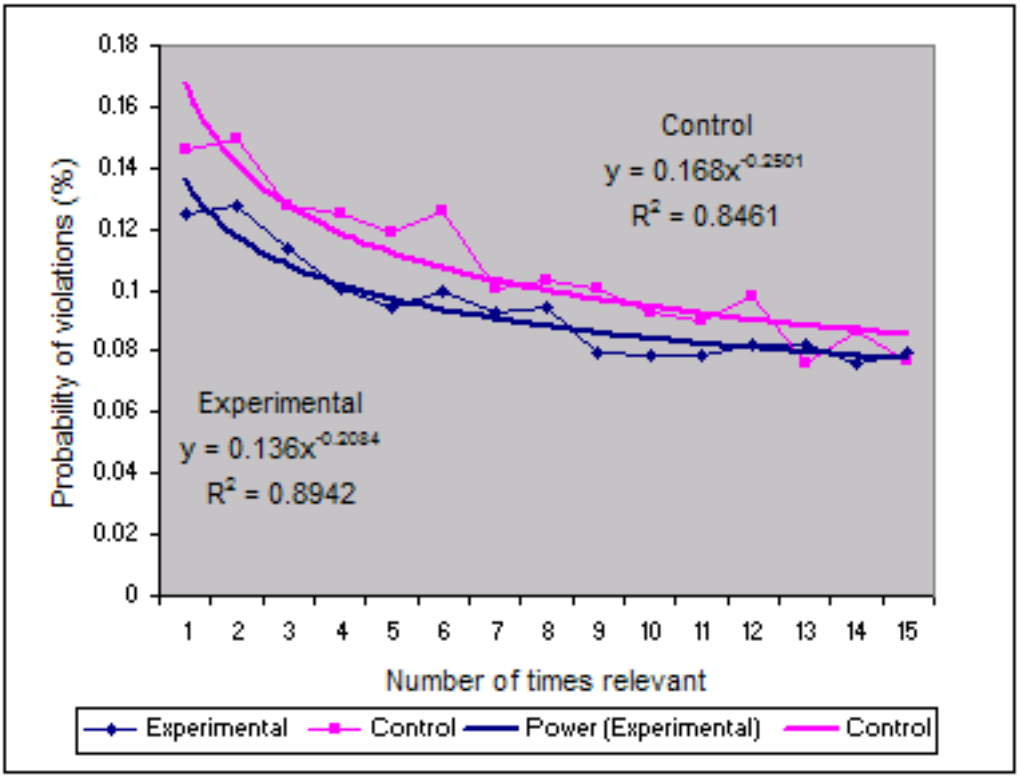

Figure 6.5: The Probability of Constraint Violations for Constraints Receiving Positive Feedback

To measure more accurately the impact of positive feedback on learning we decided to isolate only those constraints for which positive feedback was received to compare how well these constraints were learned among the two groups. We would expect to see the greatest impact of positive feedback in those constraints which directly received such feedback. To compare how well these constraints were learned by students receiving negative feedback compared to those receiving both negative and positive feedback, we plot a learning curve not of the entire population of constraints, but the subset of

constraints for which positive feedback was received. This learning curve is shown in Figure 6.5. The power curve equation for the control group was $y = 0.168x^{-0.2501}$ ($R^2 = 0.8461$) and for the experimental group it was $y = 0.136x^{-0.2084}$ ($R^2 = 0.8942$). From first glance it can be clearly seen that the experimental group learning curve is much lower than that of the control group over time. The shape of the curve can be explained as follows; the initial probability of a constraint violation in the experimental group is much lower for the experimental group while the learning rate for both groups are comparable. This result verifies that students in both groups are learning, however the reason for the difference in learning curves cannot be explained by just positive feedback.

Further analysis was attempted using pretest to separate students into categories of more able and less able based on above and below average performance on the pretest.

		Number of participants	Time	Pretest	Constraints
Control	**More**	12	232.92 (198.67)	2.5 (0.5)	11.4 (6.5)
	Less	11	151.09 (198.09)	0.9 (0.3)	8.6 (5.6)
Experimental	**More**	14	96.79 (43.61)	2.6 (0.8)	9.9 (7.3)
	Less	4	76.75 (51.62)	0.5 (0.6)	7.0 (4.2)

Table 6.8: Results Categorized According to More and Less Able Students

The overall average performance on the pretest was 1.82. 14 of the 19 students in the experimental group were classified as more able scoring above this average while 12 of the 23 students in the control group were classified as more able (see Table 6.8. We first looked at more able students for both the experimental and control group. T-test performed on time for both groups showed a significant difference between means of the groups ($p = 0.038$). More able students in the experimental group spent considerably less time than more able students in the control group. No significant difference was noted on pretest ($p = 0.92$). When we also consider less able students for each group (control and experimental) and perform a t-test on time there is no significant difference

noted ($p = 0.28$). It must be noted however that the mean time taken by less able students in the control group was less than that by the experimental group and the lack of significance may have been due to smaller numbers in the experimental group compared to the control group. The above analysis may give some evidence that positive feedback seems to be more beneficial for more able students than less able. Several other factors were analyzed, including constraints learned but no significant interactions were noted.

CHAPTER 7

Conclusions and Further Work

7.1 Conclusions

Before setting out on this journey; assessing the impact of positive feedback in constraint-based ITSs we identified clearly the motivations. It was recognized that continued research in ITSs is towards improving the effectiveness of learning within such systems. These systems provide an easily customizable, flexible, adaptable, learning sensitive and user friendly environment, where students can engage in active learning across simple and complex domains with the assistance of a coach. One of the primary goals of ITS development is maximizing the effectiveness of learning within the ITS. As much is still not known about the way in which we as humans learn, extensive research concentrates on making and evaluating conjectures of learning and instruction so as to improve ITSs. In section 3.2, we identified five hypotheses [conjectures] which were to be tested and evaluated in hopes that they would lead to more effective learning in ITSs. We also identified four objectives which we hoped would facilitate evaluation of these hypotheses.

The first of these was to study and understand positive feedback within the cognitive theory of learning. This entailed a thorough examination of associated literature

and prior research focusing on all aspects of learning, in particular learning through error detection and correction and the very limited work done on learning from correct responses. The review also included an examination of feedback in the facilitation of learning including the impacts of feedback intervention, different types of feedback content, presentation of feedback and the scheduling of feedback. Discussions were also held with experts including my supervisor Dr. Antonja Mitrović and Dr. Stellan Ohlsson of the University of Illinois. These discussions provided significant insight into not only prior research but provoked new ideas and revealed areas that could be potentially exploited to make improvements. The data used to support our hypotheses came primarily from empirical studies of human to human expert tutoring conducted at the University of Illinois and provided by their computer science department. This involved the review of analysis of hours of logged protocols based on hours of recorded data of observed tutoring by experienced human tutors to find out the pedagogical content knowledge used during tutoring. For purposes of this study it involved extraction of pedagogical knowledge used in executing positive feedback during tutoring sessions. The data supported the fact that experienced tutors do give a lot of positive feedback. The impact of the literature review was limited because literature related to the impact of positive feedback in learning was not extensive. This lead to the development of our first two hypotheses (see section 3.2); that many student steps are tentative and that providing feedback for correct answers which are in response to tentative situations, helps to reduce uncertainty and improve learning.

In order to test these two hypotheses and others, we developed a systematic approach to positive feedback based on these conjectures of positive feedback and learning; this is presented in Chapter 4. We first identified a series of situations when, based on our hypothesis, feedback should be given. This would eventually form the blueprint for implementing the scheduling mechanism for delivering positive feedback in the tutoring system. Examples of situations for which positive feedback was scheduled included giving positive feedback when the student was expressing uncertainty but nevertheless did the right thing and giving positive feedback when the student was too paralyzed

to do anything, requested assistance and in response, performed the right action. We then considered various theories and literature on feedback intervention to determine how often positive feedback should be given. It was decided that the goal should be to keep the learner focused on the actual learning of the task and in order to do so, positive feedback should not be given too often and should not be repeated sequentially for correction of the same error. A simple heuristic was developed for purposes of the study which stated that once positive feedback was given for an error, further feedback should not be given for that same error unless the student has had at least five more attempts involving that particular domain concept. We then set out the requirements for selecting and specifying the content of positive feedback. Positive feedback should contain not only verification knowledge as is common practice but more importantly should provide both topic-specific and response-specific elaboration information. This was observed during analysis of the tutoring protocols which revealed that expert tutors when providing positive feedback, not only indicate to students the correctness of their answers but also reinforce the reasons why the answer was correct. They do this either explicitly or through the use of explanation questions. This was linked directly to our third hypothesis, that positive feedback works to reinforce existing knowledge and in some cases create new knowledge. Finally in Chapter 4, we discussed the issues associated with selecting the method and form of positive feedback presentation. This included decisions such as whether positive feedback should be presented alongside negative feedback, as well as cognitive and design issues associated with developing the system's interface. The approach suggested that positive and negative feedback be given simultaneously and that the design principles of cognitive theory of multimedia learning be adhered to when designing ITS interfaces.

In Chapter 5, we described the process used to design and implement our approach to giving positive feedback using SQL-Tutor, an existing tutor used extensively at the University of Canterbury. This involved the implementation of the positive feedback scheduling mechanism which had been described generically, delivery of both verification and elaborative feedback content, and the final presentation of this feedback in

an appropriate way. The implementation of the scheduling mechanism was linked to our fourth hypothesis which stated that the constraint-base underlying student modeling in constraint-based tutors could be used to predict student uncertainty. In the previous section we identified those generic situations when positive feedback should be given. Out of this process we developed a set of feedback cases based on the constraint base of a typical constraint-based tutor. We did this using constraints and the overall student model to predict uncertainty. It was also necessary to implement cases which did not require the use of constraints. These are presented in section 5.3. The cases were successfully programmed into SQL-Tutor using functions. Verification and elaboration feedback content was generated using expert knowledge in the domain. It was also possible to include response-specific content in feedback messages using simple functions to extract from constraint bindings, student response specific information. Using the information captured from student responses we were able to tailor positive feedback to be response specific. Only very slight changes were made to the SQL-Tutor interface before the system was fully functional and we had achieved our goal of implementing our positive feedback approach in SQL-Tutor.

The final objective was achieved and is presented in Chapter 6, that is, evaluating the effectiveness of our approach to positive feedback. We conducted an evaluation study involving 55 students, the details of which we outlined in Chapter 6. The results of this study have been previously discussed but the major findings are worth repeating. We noted that the experimental group, the group using our implemented approach to positive feedback, performed better than the control group. They solved almost the same number of problems in significantly less time, making fewer attempts while attempting almost the same number of problems. Statistical analysis showed not only time to be significant but the average time per solved problem as well as the average time per problem. This significant difference in time supports the hypothesis that positive feedback helps to reduce student uncertainty and by so doing reduces both the time taken for students to solve problems and also the number of errors. This also goes to support the hypothesis that many student steps are indeed tentative surrounded by a cloud of uncer-

tainty. Positive feedback was also noted to be a significant factor in determining how much students are learning, accounting for 6% of the variance noted in student learning; also students have learnt almost the same number of constraints in significantly less time. This stage supported all our hypotheses with the exception that positive feedback would, at least in the short-term, result in higher learning rates. It was noticed that the learning rate, measured as the number of constraint violations per number of times a constraint is relevant, was slightly higher for the control group. Our final hypothesis regarding improved learning rates is therefore not supported.

We conclude that positive feedback does improve the effectiveness of learning in Intelligent Tutoring Systems, decreasing the amount of time required to see the benefits of using such systems and as such, positive feedback results in increased amount of learning over a shorter period of time.

7.2 Further Work

The contribution of this research fits into the overall work being done with regards to the role of feedback in the acquisition of skills and learning in general. Typically feedback on correct responses has been viewed or implemented as as a way of telling the learner that their understanding of the material being presented is sufficient and often flags the level of their competence and achievement of goals. In this research we have taken this approach further, providing topic-specific and response-specific elaborative information in feedback messages while implementing feedback scheduling based on the notion of student uncertainty. Further work will involve firstly addressing some of the problems discovered during our evaluation study and secondly making improvements to the overall system to see what further improvements can be made to learning. Under the former we will have to correct the problem discovered with the encoding of pre/post test answers. Recall that the problem resulted in no reliance being placed on post-test scores and hence we were unable to measure with any accuracy, learning gains. Once this has been done, additional studies will have to be conducted and further analysis done. It will

also be necessary to conduct studies within other domains to ensure that any improvements in learning are domain transferable. Given the results of these are promising, we propose implementing a series of extensions, outlined below, which will hopefully lead to the improvement of the system.

Positive Feedback Content Generation Fist we consider the area of feedback content generation. In this research we have generated content using a combination of pre compiled messages produced by experts in the domain and response-specific information obtained from constraint bindings via student responses. In some cases, we have re-stated the constraint in general terms e.g. "That is correct. When C_r is the case, C_s has to be the case as well." In other cases we have re-stated the constraint as instantiated vis-à-vis the problem situation which involved grabbing the variable bindings from variables in the constraint and translating these into English.

We propose the following extension and improvement for positive feedback content. This is based on stating alternatives and why it is they are or are not correct. We propose that positive feedback content include a statement about how alternative actions that could have been considered would have violated the given constraint. One proposal is to include a statement of the general form: "Very good! X is the right thing to do here, because if you had done Y instead, you would have gotten [description in here], and as you know, when C_r is the case, C_s had better be the case too." and once again this can be instantiated vis-à-vis the problem situation.

Further tailoring of content to sound more conversational or dialogue like could be explored as it has been shown the tutors which produce better dialogue also produce better learning.

Positive Feedback Scheduling The feedback scheduling mechanism presented in this research is built primarily on the premise that most of students actions are tentative and that the student is either guessing or is at least exhibiting some degree of uncertainty about what it is they should do. This research takes advantage of the information pro-

vided through constraints and uses this student modeling data to predict uncertainty. Several cases have been implemented using this approach but admittedly this may not be exhaustive. Further work can be done exploring new cases and tweaking existing cases to measure any improvements or gains in learning.

Alternatively other measures of uncertainty can be tested. Proposed measures include experimenting with response times to determine moments of hesitation or having students themselves rate their uncertainty level on some type of rating scale placing reliance on their degree of self awareness. More complex techniques including the implementation of fuzzy logic [Kavcic 2002] or a combination of measures can also be tested. If uncertainty can be more accurately and reliably predicted then so would the timing of positive feedback interventions and it would be expected that learning effects would improve as a result.

Presentation of Positive Feedback Only very minor changes were made to the interface of SQL-Tutor and no changes were made to the mode of presentation of feedback messages within the system. Recall that we proposed in Section 4.3, the use of acoustic and verbal presentation methods to reduce single channel overload. Positive feedback was not presented using acoustic methods as was encouraged by the cognitive theory of multimedia learning. Feedback content was presented as black text on a gray background in both versions of the system. It would be interesting to measure the influence of various presentation media on positive feedback. These may include verbal presentation or pictorial presentations depending on the domain in question and applying the principles of multimedia design. It is our hope that this can be implemented as part of another study and a further extension of the system.

This study has shown that positive feedback if implemented correctly can lead to significant improvement in learning and the ideas presented above can lead to further improvements in the approach to positive feedback that has been presented.

CHAPTER 8

References

A.C. Graesser, N. Person, D. H. and TRG [2001], 'Teaching tactics and dialog in autotutor', *International Journal of Artificial Intelligence in Education* (12), 257–279.

Anderson, J. R. [1983], *The Architecture of Cognition*, Harvard University Press, Cambridge MA.

Anderson, J. R. and Corbett, A. T. [1994], 'Knowledge tracing: Modeling the acquisition of procedural knowledge', *User Modeling and User-Adapted Interaction* **4**(4), 253–278.

Anderson, J. R., Corbett, A. T., Koedinger, K. R. and Pelletier, R. [1995], 'Cognitive tutors: Lessons learned', *The Journal of the Learning Sciences* **4**(2), 167–207.

Azevedo, R. and Bernard, R. M. [1995*a*], 'A meta-analysis of the effects of feedback in computer-based instruction', *Journal of Educational Computing Research* **13**(2), 111–127.

Azevedo, R. and Bernard, R. M. [1995*b*], 'A meta-analysis of the effects of feedback in computer-based instruction', *Journal of Educational Computing Research* **13**(2), 111–127.

Beck, J., Stern, M. and Haugsjaa, E. [1996], 'Applications of ai in education', *Crossroads. Special issue on artificial intelligence* **3**(1), 11–15.

Bloom, B. [1984], 'The 2 sigma problem: The search for methods of group instrcutiion as effective as one-to-one tutoring', *Educational Researcher* **13**, 3–16.

Choquet, C., Danna, F., Tchounikine, P. and Trichet, F. [1998], Modelling the knowledge-based components of a learning environment within the task/method paradigm, *in* 'Proceedings of the 4th International Conference on Intelligent Tutoring Systems', number 1452 *in* 'Lecture Notes in Computer Science', Springer-Verlag, London UK, pp. 56–65.

Core, M. G., Moore, J. D., Zinn, C. and Wiemer-Hastings, P. [2000], Modeling human teaching tactics in a computer tutor, *in* 'Proceedings of the ITS Workshop on Modeling Human Teaching Tactics and Strategies', Montreal Canada.

Elmasri, R. and Navathe, S. [2007], *Fundamentals of Database Systems (5th edition)*, Addison-Wesley.

Eugenio, B. D., Lu, X., Kershaw, T. C., Corrigan-Halpern, A. and Ohlsson, S. [2005], Poster reception: Positive and negative verbal feedback for intelligent tutoring systems, *in* '12th International Conference on Artificial Intelligence in Education', Amsterdam Netherlands.

Frasson, C., Mengelle, T., Aimeur, E. and Gouardéres, G. [1996], An actor based architecture for intelligent tutoring systems, *in* 'Proceedings of the Third International Conference on Intelligent Tutoring Systems', number 1086 *in* 'Lecture Notes in Computer Science', Springer-Verlag, London UK, pp. 57–65.

Freedman, R. [1999], Atlas: A plan manager for mixed-initiative, multimodal dialogue, *in* 'AAAI-99 Workshop on Mixed-Intiative Intelligence', Orlando.

Gertner, A. and VanLehn, K. [2000], Andes: A coached problem solving environment for physics, *in* 'Gauthier, G., Frasson, C. and VanLehn, K. (Eds) Intelligent Tutoring Systems: 5th International Conference, ITS 2000', pp. 131–142.

Gilman, D. A. [1969], 'Comparison of several feedback methods for correcting errors by computer-assisted instruction', *Journal of Educational Psychology* **60**(6), 503–508.

Hartley, R., Paiva, A. and Self, J. [1995], Extemalising learner models, *in* 'Proceedings of the International Conference on Artificial Intelligence in Education', Washington, pp. 509–516.

Heffernan, N. T. [2001], Intelligent tutoring systems have forgotten the tutor: Adding a cognitive model of human tutors, Wishful Research Result CMU-CS-01-127, Carnegie Mellon University, School of Computer Science, Pittsburgh.

Heffernan, N. T. and Koedinger, K. R. [2002], An intelligent tutoring system incorporating a model of an experiences human tutor, *in* 'Sixth International Conference on Intelligent Tutoring System', Biarritz France.

K. Zakharov, A. Mitrovic, S. O. [2005], Feedback micro-engineering in eer-tutor, *in* 'C-K Looi, G. McCalla, B. Bredeweg, J. Breuker (eds) Proc. Artificial Intelligence in Education', IOS Press, Amsterdam, Netherlands, pp. 718–725.

Katz, S., Lesgold, A., Peters, D., Eggan, G. and Gordin, M. [1998], Sherlock 2: An intelligent tutoring system built on the lrdc framework, *in* 'C.P. Bloom and R.B. Loftin (Eds.), Facilitating the development and use of interactive learning environments', Hillsdale, NJ, pp. 227–258.

Kavcic, A. [2002], 'Dealing with uncertainty of user knowledge: Fuzzy user modelling in educational hypermedia'. Invited Lecture: Advances in Multimedia Communications, Information Processing and Education.

Kluger, A. N. and Denisi, A. [1996], 'The effects of feedback interventions on performance: A historical review, a meta-analysis, and a preliminary feedback intervention theory', *Psychological Bulletin* **119**(2).
URL: *http://proxies.apa.org/journals/bul/119/2/254*

Koedinger, K. R. [1998], Intelligent cognitive tutors as modeling tool and instructional model, Position paper for the nctm standards 2000 technology conference, Carnegie Mellon University, Pittsburgh Advanced Cognitive Tutor (PACT) Center, Human-Computer Interaction Institute, School of Computer Science. Downloadable from www.cogsci.northwestern.edu.

Kulhavy, R. W. and Stock, W. A. [1989], 'Feedback in written instruction: The place of response certitude', *Educational Psychology Review* **1**(4), 279–308.

Kulik, J. A. and Kulik, C. C. [1988], 'Timing of feedback and verbal learning', *Review of Educational Research* **58**(1), 79–97.

Lajoie, S. and Lesgold, A. [1989], 'Apprenticeship training in the workplace: Computer-coached practice environment as a new form of apprenticeship', *Machine-Mediated Learning* **3**(1), 7–28.

Lu, X. [2006], Expert tutoring and natural language feedback in intelligent tutoring systems, *in* 'Doctoral Student Consortium at the 14th International Conference on Computers in Education', Beijing China.

Martin, J. and VanLehn, K. [1995], 'Student assessment using bayesian nets'. Joel Martin and Kurt VanLehn. Student assessment using Bayesian nets. International Journal of Human-Computer Studies, 42:575–591, 1995.
URL: *citeseer.ist.psu.edu/martin95student.html*

Mason, B. J. and Bruning, R. [2001], Providing feedback in computer-based instruction: What the research tells us, Dissertation, University of Nebraska, Center for Instructional Innovation, Lincoln, Nebraska.

Mathews, M. [2006], Investigating the effectiveness of problem templates on learning in intelligent tutoring systems, Bsc honors report, University of Canterbury, Department of Computer Science and Software Engineering, Christchurch, New Zealand.

Mayer, R. E. [2001], *Multimedia Learning*, Cambridge University Press.

Mayer, R. E. [2002], 'Cognitive theory and the design of multimedia instruction: An example of the two-way street between cognition and instruction', *New Directions for Teaching and Learning* **2002**(89), 55–71.

Mitrovic, A. [1998], A knowledge-based teaching system for sql, *in* T. Ottmann and I. Tomek, eds, 'Proceedings of ED-MEDIA/ED-TELECOM'98', Association for the Advancement of Computing in Education, Freiburg, pp. 1027–1032.

Mitrovic, A. [2003], 'An intelligent sql tutor on the web', *International Journal of Artificial Intelligence in Education* **13**, 171–195.

Mitrovic, A. and Martin, B. [2004], 'Evaluation intelligent education systems with learning curves'. presented at The Workshop on Evaluation at Adaptive Hypermedia 2004.

Mitrovic, A. and Martin, B. [2005], 'Using learning curves to mine student models'. presented at The 10th International Conference on User Modelling UM2005.

Mitrovic, A., Mayo, M., Suraweera, P. and Martin, B. [2001], Constraint-based tutors: a success story, *in* L. Monostori, J. Vancza and M. Ali, eds, 'Proceedings of 14th Conference on Industrial and Engineering Applications for Artificial Intelligence and Expert Systems IEA/AIE-2001', Springer-Verlag Berlin Heidelberg LNAI 2070, Budapest, pp. 931–940.

Mitrovic, A. and Ohlsson, S. [1999], 'Evaluation of a constraint-based tutor for a database language', *International J. Artificial Intelligence in Education* **10**(3-4), 238–256.

Moreno, R. and Mayer, R. [2000], Meaningful design for meaningful learning: Applying cognitive theory to multimedia explanations, *in* 'ED-MEDIA 2000 Proceedings', pp. 747–752.

Moreno, R. and Mayer, R. E. [1997], 'A cognitive theory of multimedia learning: implications for design principles'. See http://www.unm.edu.

Mory, E. H. [1995], 'Feedback and self-regulated learning: A theoretical synthesis', *Review of Educational Research* **65**(3), 245–281.

Narciss, S. and Huth, K. [2004], How to design informative tutoring feedback for multimedia learning, *in* H. M. Niegemann, R. Brnken and D. Leutner, eds, 'Instructional design for multimedia learning', Munster: Waxman, Germany, pp. 181–195.

Nkambou, R. and Gauthier, G. [1996], 'Integrating www resources in an intelligent tutoring system', *Journal of Network and Computer Science Application* **19**(4).

Nkambou, R. and Kabanza, F. [2001], 'Designing intelligent tutoring systems: a multiagent planning approach', *ACM SIGUE Outlook* **27**(2), 46–60.

Ohlsson, J. [2004], 'Instrumental knowledge and practice'. Learning and Instruction: An Introduction (draft), Department of Psychology, University of Illinois.

Ohlsson, S. [1993], 'Constraint-based student modeling', *Journal of Artificial Intelligence in Education* **3**(4), 429.

Ohlsson, S. [1994], Constraint-based student modelling, *in* 'Proceedings of Student Modelling: the Key to Individualized Knowledge-based Instruction', Springer-Verlag, Berlin, pp. 167–189.

Ohlsson, S. [1996], 'Learning from performance errors', *Psychological Review* **103**(2), 241–262.

Ohlsson, S. [2006], Positive feedback. Email Communication with Stellan Ohlsson.

Ohlsson, S. [2007], Positive feedback. Email Communication with Stellan Ohlsson.

Ohlsson, S., Eugenio, B. D., Chow, B., Fossati, D., Lu, X. and Kershaw, T. [2007], Beyond code-and-count analysis of tutoring dialogues, *in* R. Luckin, K. R. Koedinger and J. greer, eds, 'Artificial Intelligence in Education: Building Technology Rich Learning Contexts That Work', IOS Press, Amsterdam, Netherlands, pp. 349–356.

Ong, J. and Ramachandran, S. [2003], Intelligent tutoring systems: Using ai to improve training performance and roi, Technical report, Stottler Henke Associates.

Paivio, A. [1986], *Mental representations: a dual coding approach*, Oxford University Press, Oxford, England.

Pillay, N. [2003], 'Developing intelligent programming tutors for novice programmers', *ACM SIGCSE Bulletin* **35**(2), 78–82.

Planty, M., Provasnik, S. and Hussar, W. [1999], The condition of education, Nces publication no. 1999-022, National Center for Education Statistics, Office of Educational Research and Improvement, Department of Education, Washington, DC.

Ritter, S., Brusilovsky, P. and Medvedeva, O. [1998], Creating more versatile intelligent learning environments with a component-based architecture, *in* 'Proceedings of the 4th International Conference on Intelligent Tutoring Systems', number 1452 *in* 'Lecture Notes in Computer Science', Springer-Verlag, London UK, pp. 554–563.

Schooler, L. and Anderson, J. [1990], The disruptive potential of immediate feedback, *in* 'Proceedings of the Twelfth Annual Conference of the Cognitive Science Society', Association for the Advancement of Computing in Education, Cambridge, pp. 702–708.

Schulze, K. G., Shelby, R. N., Treacy, D. J., Wintersgill, M. C., Vanlehn, K. and Gertner, A. [2000], 'Andes: An intelligent tutor for classical physics', *The Journal of Electronic Publishing* **6**(1), 78–82.

Skinner, B. F. [1968], *The Technology of Teaching*, B. F. Skinner Foundation, New York: Appleton-Crofts.

Stauffer, K. [1996], Student modeling and web-based learning systems, Short research project, Athabasca University.

Thorndike, E. L. [1913], *The original nature of man*, second edn, Teachers College, Columbia University, New York.

Tsybenko, Y. [1995], "device models" in student modeling, *in* 'Proceedings of the International Conference on Artificial Intelligence in Education', Washington, pp. 525–532.

Vanlehn, K., Jones, R. M. and Chi, M. T. H. [1992], 'A model of the self-explanation effect', *Journal of the Learning Sciences* **2**(1), 1–59.

Whyte, M. M., Karolick, D. M., Neilsen, M. C., Elder, G. D. and Hawley, W. T. [1995], 'Cognitive styles and feedback in computer-assisted instruction', *Journal of Educational Computing Research* **12**(2), 195–203.

CHAPTER 9

Appendices

9.1 Appendix A: Calculating Learned Constraints

We outline below the heuristic used in the calculation of learned constraints or what is sometimes referred to as the knowledge score. This method has previously been tested having been used in other studies of SQL-Tutor. It uses a floating window, in this case of size 5, to ascertain the relevant section of the students knowledge-component history that is to be considered. The measure of learned constraints is a score given as a probability (0-1) that a knowledge-component (constraint) has been learnt and is used to facilitate pedagogical decisions based on the students current knowledge level.

The score is calculated as follows: each constraint (knowledge-component) is considered individually, initially taking the first five (window size) attempts involving the constraint. If the student satisfies the constraint less than seventy percent (70%) of the number of times it is relevant, then the student is recorded as knowing the constraint otherwise the constraint is flagged as a constraint to be learnt. Seventy percent (0.7) is referred to as the tolerance level and this represents the threshold probability for determining a learned constraint. At the end of the session the last five attempts are taken and the same calculation performed. If the student previously did not know the

constraint but has now satisfied the constraint more than seventy percent (70%) of the number of times its relevant, then the constraint has been learnt. The measure therefore only applies to newly acquired constraints or concepts, that is, those learned as a result of using the system. Constraints which have a history of less than 5 attempts are not considered. Consider the example below. The constraint history shown consist of zeros (constraint was violated) and ones (constraint was used correctly).

Example 1

Knowledge-component history: (0 0 1 1 0 1 0 1 1)
Initial Score: 2/5
Final Score: 3/5
Result: Constraint not learnt

Example 2

Knowledge-component history: (0 0 1 0 1 1 1 1)
Initial Score: 2/5
Final Score: 4/5
Result: Constraint learnt

9.2 Appendix B: Pre- and Post-tests

The following tests were used in the evaluation study of SQL-Tutor.

Version 1

Question 1. What clause of the SELECT statement allows tuples to be selected?

a. SELECT
b. FROM
c. WHERE
d. GROUP BY
e. HAVING
f. ORDER BY

Question 2. All attributes listed in the ORDER BY clause of a SELECT statement must also appear in the SELECT clause. (T/F)

Question 3. We need to find the titles of all movies other than comedies. The following SQL statement achieves that. (T/F)

```
select TITLE
from MOVIE
where TYPE = 'comedy' or 'drama';
```

Question 4. Which of the following would allow all tuples of table R to be kept in the resulting table, but only those tuples from S that have matching tuples in R?

a. from R join S on R.A=S.B
b. from R right outer join S on R.A=S.B
c. from S left outer join R on R.A=S.B
d. from R left outer join S on R.A=S.B
e. from R full outer join S on R.A=S.B
f. from R inner join S on R.A=S.B

Version 2

Question 1. What clause of the SELECT statement allows the resulting table to be sorted?

a. SELECT
b. FROM
c. WHERE
d. GROUP BY
e. HAVING
f. ORDER BY

Question 2. Attribute names used in subqueries are assumed to come from tables used in the outer query. (T/F)

Question 3. A SELECT statement contains a nested query in the WHERE clause, comparing the value of an attribute to the values returned by the nested SELECT with an IN predicate. Which of the following predicates can be used in that statement instead of IN?

a. = ANY
b. = ALL
c. <> ALL
d. <> ANY
e. = EVERY

Question 4. We need to find the titles of all movies other than comedies. The following statement will achieve that. (T/F)

```
SELECT TITLE
FROM MOVIE
WHERE TYPE = NOT('comedy')
```

9.3 Appendix C: Problem Selection

The student is presented with the choice of SQL clauses, with the system choice highlighted and selected (see Figure 9.1).

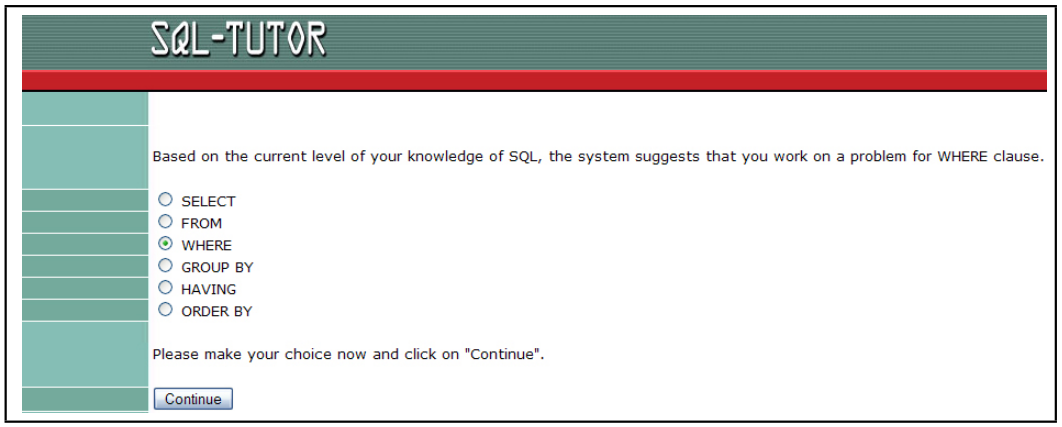

Figure 9.1

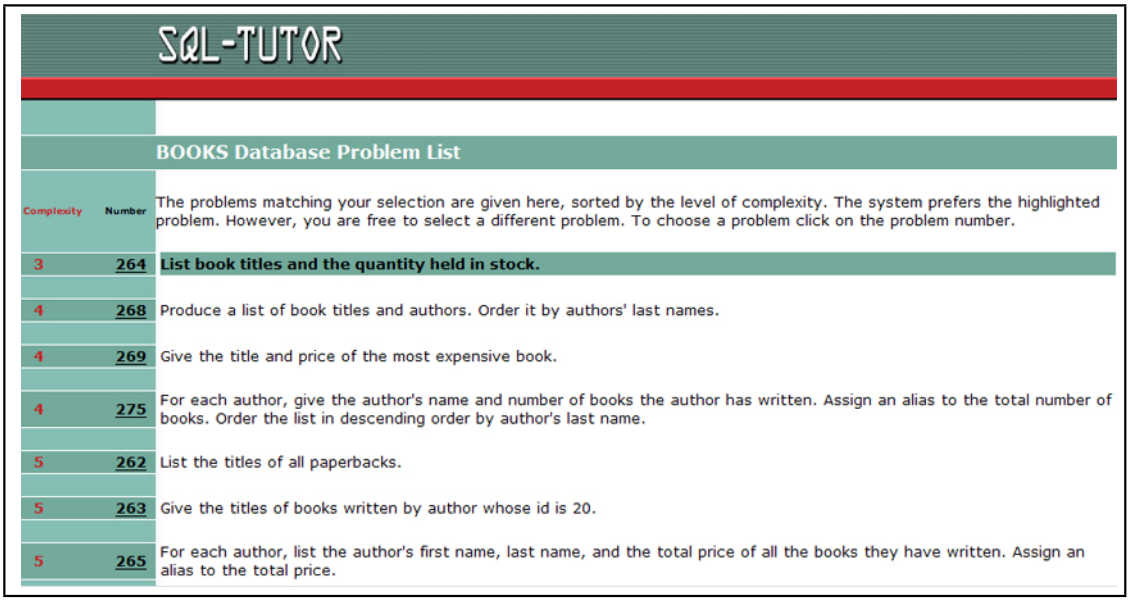

Figure 9.2

Once the student chooses the SQL clause, a list of problems testing the clause is displayed with the system choice highlighted (See Figure 9.2).

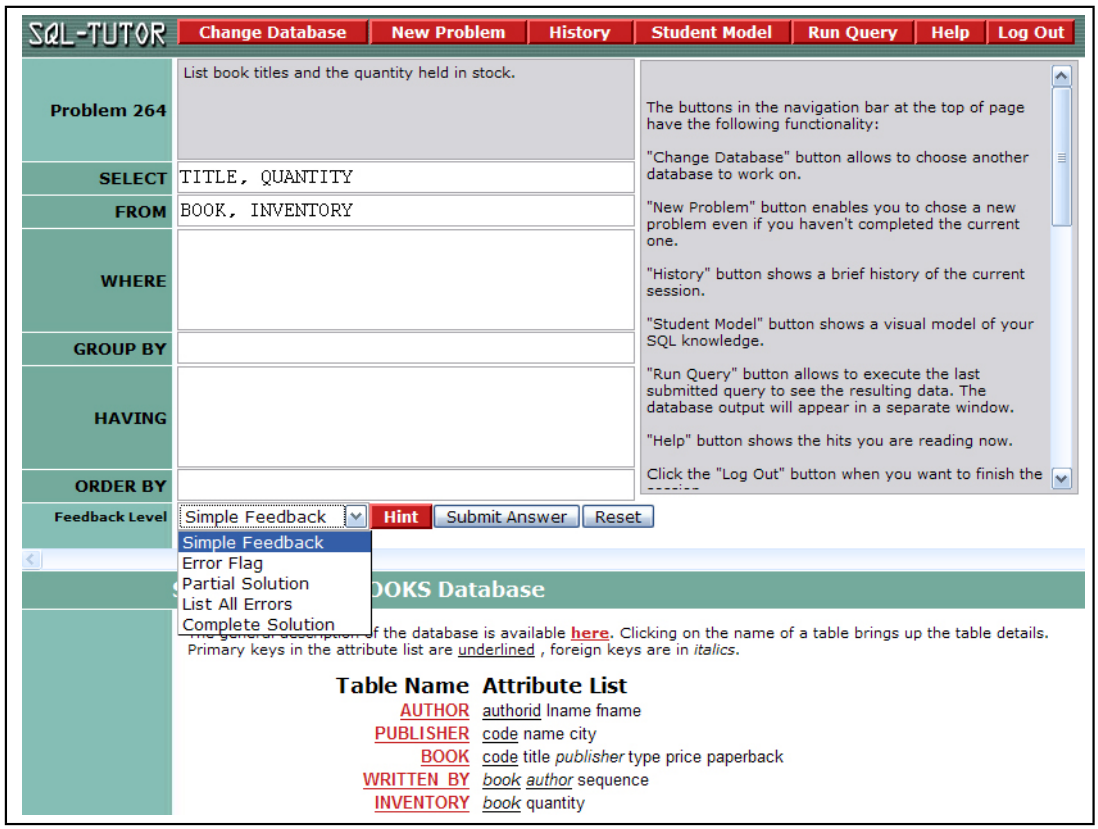

Figure 9.3

After making a selection from the problem list, the problem, solution space, feedback area, and associated information are displayed in the task environment (see Figure 9.3). The student can attempt to solve the problem, request feedback, request, help, view their student model, view their history, or run their query on a sample database.